POLITECNICO DI TORINO
Ingeneria Industriale (Specialità Elettronica e Automatica)
Dipartimento di Elettronica e Telecomunicazioni (DET)

TITLE: Arduino –based device for voice monitoring.
TITOLO: Dispositivo basato su Arduino per il monitoraggio della voce.

Laboratorio di misure elettriche ed elettroniche
del Politecnico di Torino.

Relatore Politecnico di Torino: Ing. Alberto Vallan
Relatore Politecnico di Torino: Dott. Alessio Carullo
Correlatore UC3M: Juan Carlos González Victores

Tesi di Laurea di:
Sara Ruiz Iglesias

Anno Accademico 2014-2015

A tutti quelli che
mi hanno dato sopporto
e sono stati al mio fianco
soprattutto questo ultimo anno...

Indice

Indice di immagini

SOMARIO

In questo documento si presenta la continuazione dello sviluppo di un dispositivo basato su Arduino per il monitoraggio della voce. Fa uso di due microfono, uno d'aria ed altro di contatto o di gola collegati come canali di ingresso alla scheda Arduino. Ha anche altre due componenti: un sensore di temperatura e umidità per la caratterizzazione dell'ambiente in cui si realizza il monitoraggio e un paio di amplificatori programmabili per aumentare la risoluzione dei segnali di ingresso.

L'obiettivo principale del dispositivo è la caratterizzazione della voce mediante il calcolo dei parametri fondamentali. L'obiettivo secondario è che questi parametri possono servire a diversi studi futuri nell'area clinica come: cercare di individuare i rischi di essere sotto un disturbo della voce, monitorizare persone con le corde vocali sane e persone con disturbi per trovare una relazione tra le loro abitudini e i disturbi... Per questa ragione è un dispositivo pensato per essere utilizzato durante tutto il giorno soprattutto per persone che lavorano molto con la voce come professori e cantanti. Questa possibilità di fare il monitoraggio fuori dell'ambiente clinico e con una durata così lunga fornisce dati più reali.

Il funzionamento del dispositivo è il monitorare la voce mediante campionamento continuo delle due segnali e calcolare i parametri in tempo reale, caratterizzare l'ambiente (rumore di fondo e temperatura e umidità), e analisi posteriore più in profundità con l'informazione salvata nella scheda SD.

Ha due modi di funzionamento, uno di calibrazione ed un altro di monitoraggio propriamente. Il primo serve a calibrare i microfoni e verificare il corretto funzionamento dopo avere attaccato tutto e il secondo a realizzare il monitoraggio.

Il lavoro consiste nella implementazione del dispositivo, la realizzazione dei test di controllo del suo corretto parcialmente e totalmente, e di una prova di monitoraggio che è l'obiettivo del lavoro.

ABSTRACT

This document is the development of an Arduino-based device for voice monitoring. It uses two microphones, a contact microphone and an air one, connected as input channels. It also uses another two components: a humidity and temperature sensor to characterize the environment; and a pair of programmable amplifiers to have a better signal resolution.

The main goal is voice characterization by fundamental parameters calculation. The second goal is providing these parameters for future clinical studies, such as finding the risk of being under a voice disturb, voice monitoring of people with voice disturbs and without them, finding a relation between behaviors and disturbs… Because of that, it's a device thought to be used during the daily work and at home, mainly by people that makes extensive use of their voices like professors and singers. This possibility of voice monitoring out of the clinical environment and during a whole day provides more realistic data.

Device functioning is voice monitoring by continuous sampling and parameters calculation in real time, environment characterization (background noise, temperature and humidity), and further analysis with information saved in SD card.

It has two operating modes, calibration and monitoring. The first one is used for microphones calibration and operating verification after connecting all the components and the second one is used for monitoring.

This work comprises device implementation, partial and total verification test and a final monitoring test.

1. INTRODUZIONE

Il lavoro svolto a continuazione e lo sviluppo di un dispositivo per monitorizare diversi parametri della voce umana. Questi parametri sono stati scelti per due ragioni: permettono caraterizzare la voce e anche di avere un'informazione sulla possibilità di essere in rischio di avere delle malattie relative al sistema vocale del corpo umano. Si tratta di un dispositivo mobile con grande autonomia pensato per essere utilizzato per i professionisti che fano uso della voce nel suo lavoro come professori, cantanti…

I parametri studiati sono: la frequenza naturale (F0) e la Vrms (root mean square voltage o valore efficace di tensione). L'obiettivo di questa tesi è la caratterizzazione della voce mediante questi due parametri. Per capire perchè si hanno scelto e come è possibile misurarli, si deve capire prima come lavora il sistema vocale e come si genera il suono e la voce.

1.1 Generazione e propagazione del suono

L'obiettivo di essere capaci di sapere quando c'è il rischio di soffrire una malattia si raggiunge col analisi delle caratteristiche sonore della voce. , per questo, la prima cosa da capire è la generazione e propagazione del suono.

1.1.1 Generazione del suono:

Il suono si define per la ANSI/ASA (American National Standards Institute (ANSI) on Acoustical Terminology) come ""(a) Oscillation in pressure, stress, particle displacement, particle velocity, etc., propagated in a medium with internal forces (e.g., elastic or viscous), or the superposition of such propagated oscillation. (b) Auditory sensation evoked by the oscillation described in (a)"[14], cioè la oscillazione del livello di presione, stress, displazamento di particole, ecc propagato in un medio con forze interne (per esempio elastiche o viscose) o la superposizione di queste propagazioni delle oscillazioni.

In questo caso, le oscillazioni che generano il suono si propagano in aria e si generano quando l'aria dei pulmoni passa attraverso delle corde vocali con una apertura maggiore oppure minore. I diversi suoni generati come vocali o consonanti sono modulati per le diverse configurazioni della cavità orale. Questi suoni sono funzione dell'apertura della bocca, la posizione dei denti, della lingua, delle labbra...

Esempi di sorgenti di suono:

La generazione del sono si produce per il disturbo del equilibrio della densità o della pressione in un gas, liquido o solido (in questo caso, un gas, l'aria), la propagazione di questa perturbazione è la causa delle oscillazioni di cui si ha parlato prima.

Gli esempi tipici per spiegare le perturbazioni che generano suono, sono un pistone in un cilindro, un battimani, e una sirena. Questi ultimi con un interesse speziale sul come si genera il suono nel corpo umano come si verrà più avanti.

Nel caso del esempio del pistone nel cilindro, il movimento ciclico del cilindro altera la densità delle particelle d'aria al interno. Le particelle in contatto col cilindro acquistano questo stesso movimento e lo trasmettono alle particelle vicine per collisione e così si propaga la perturbazione di pressione nel medio. Questa propagazione della perturbazione è la cosidetta propagazione del suono, la propagazione dei movimenti oscillatori delle particelle è su se stessi, è la perturbazione che si propaga. Le particelle

acquistano un movimento oscillatorio dove la posizione d'equilibrio è quella che avevano prima che arrivasse il disturbo. Quando si produce la collisione tra di loro la particella ch'era in riposo aumenta la sua energia cinetica, mentre l'energia dell'altra diminuisce per la conservazione della energia che in questo caso si spiega con la conservazione del momento cinetico(m·v), che si mantiene costante nelle collisioni.

E così viene trasmessa la perturbazione. Una volta che passa la perturbazione, l'ampiezza di questo movimento oscillatorio diminuisce finché la particola ritorna allo stato di equilibrio.

Questo esempio è molto utile per capire la generazione e sopratutto la propagazione del suono. I seguenti esempi serviranno a capire meglio questo processo nella produzione di suono, della voce, per il corpo umano.

Nel esempio del battimani, la perturbazione che genera il suono è l'interruzione del flusso d'aria, strizzato tra le mani. Questa perturbazione puntuale può essere amplificata se la posizione delle mani viene modificata in modo che lo spazio tra loro si comporti come una camera di risonanza come accade con la cavità orale. La generazione di alcuni suoni fatti per esempio con la lingua contro il palato assomigliano il processo di generazione del suono che ha luogo nel battimani.

La generazione di suono che accade in una sirena, invece, serve a spiegare il processo di generazione del suono per la glottide, perché sono processi simili. In questo esempio si ha un getto d'aria contro i buchi fatti nel perimetro di una ruota che gira. La perturbazione che genera il suono accade di forma periodica quando il flusso d'aria che attraversa i buchi viene interrotto. Più veloce è il giro della ruota, più alta è la frequenza delle interruzioni del flusso d'aria e più alte le componenti di frequenza del suono generato. La sirena e una sorgente interessante perché assomiglia alla glottide. In entrambi casi l'aria attraversa orifizi che si aprono e chiudono di forma periodica e che hanno come risultato una serie di pulsi d'aria.

Questa somiglianza è più forte se invece di fare la comparazione con uno degli esempi, si fa con la combinazione di entrambi, se si considera la vibrazione delle corde vocali come una combinazione di una sirena e un battimani distinguendo così tra spostamento del flusso d'aria in tutte le direzione, come accade nel esempio del battimani, a causa della spiegata interruzione del flusso e il flusso attraverso la glottide analogo al flusso attraverso i buchi del esempio della sirena.

1.2 Anatomia basica del sistema vocale umano:

Una volta capito il processo fisico che occorre per la generazione del suono, bisogna capire anche, la fisiologia e funzionamento del corpo umano che genera, propaga e modula il suono per produrre la voce.

Nella generazione del suono per l'essere umano, nella generazione della voce, interviene il sistema respiratorio, composto per il tratto respiratorio superiore e il tratto respiratorio inferiore come si vede nel schema della figura 1.

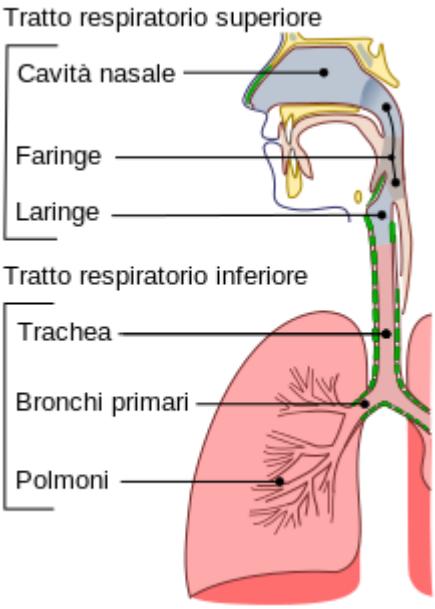


Figura 1: Tratto respiratorio umano

L'organo che si può considerare il più importante e la laringe, ma per capire il suo funzionamento, bisogna fare attenzione a tutto il sistema respiratorio perché la maggior parte dei movimenti della laringe sono riflessivi cioè, si fanno senza pensare e vengono determinati anche per il funzionamento di altre organi.

La posizione della laringe può variare verticalmente e orizzontalmente. Nella deglutizione, i muscoli forzano alla laringe a muoversi verso l'alto per chiudere il condotto della respirazione, mentre nello sbadiglio, la faringe si espande e forza la laringe a muoversi per ampliare il condotto dell'aria. In questi casi, la laringe può muoversi alcuni centimetri verso l'alto oppure verso il basso.

Può anche muoversi avanti e indietro per esempio se c'è bisogno di un gran flusso d'aria quando un bolo alimentare, entra nell'esofago. Questa capacità che deve avere la laringe spiega perché la maggior parte della sua struttura è composta per cartilagini, solo c'è un osso vicino ed è un osso flottante.

L'informazione delle seguente figure permette una migliore comprensione del processo di generazione della voce e anche del processo di campionamento e le caratteristiche dei campioni.

Nella figura 2 si può vedere uno schema più completo del tratto respiratorio superiore. In questa immagine si mostrano le posizioni di componenti importanti nella modulazione dei suoni com'è la cavità orale e la lingua e anche la posizione dei diversi cartilagini e muscoli rispetto dell'esterno del corpo. Sapere questa posizione dallo sterno e molto importante in questo caso per scegliere la posizione ottima dove mettere il microfono a contatto per avere dei campioni adeguata.

Guardando nelle immagine 2 e 3 la posizione della "Laryngeal prominence", detta comunemente mela di Adamo, appartenente alla cartilagine tiroidea si ha una idea della situazione del resto dei cartilagini e muscoli nel collo.

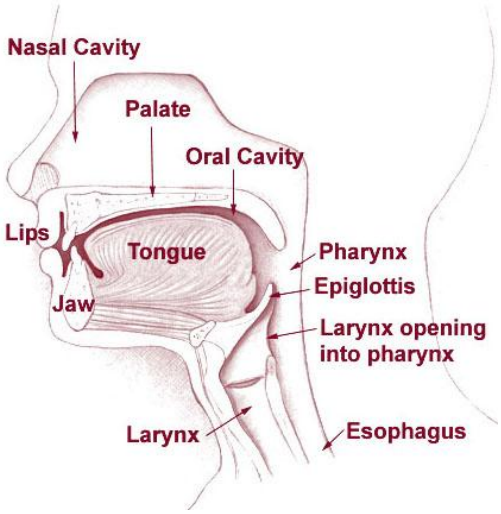
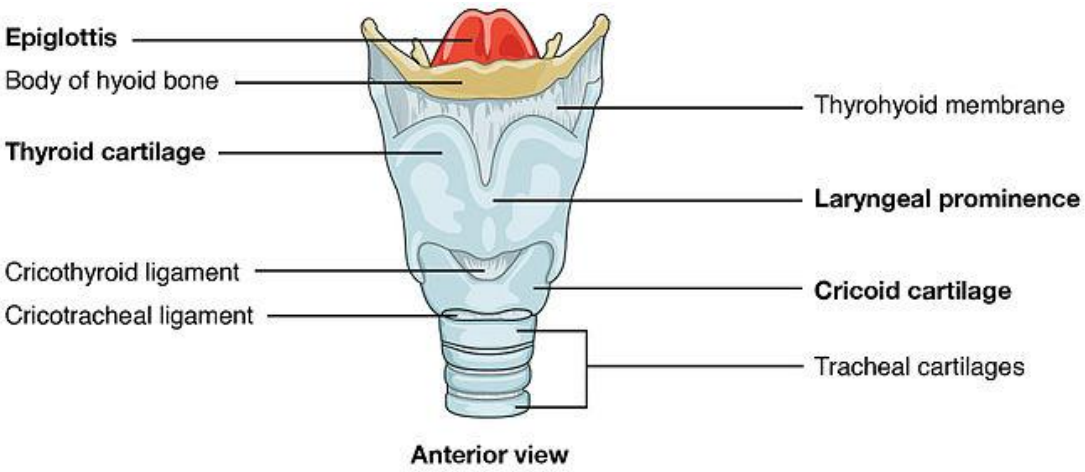


Figura 2: Tratto respiratorio superiore

Nelle due immagine compressi nella figura 3, si può vedere la fisiologia della laringe, con vista frontale e laterale. Grazie a queste immagine, si può vedere esattamente dove sono le corde vocali rispetto all'esterno e rispetto al condotto della laringe. Lo spazio tra le corde vocale si chiama glottide e nella anatomia del parlato, è un punto di riferimento importante. per questa ragione si define:

Subglottico: Sotto la glottide.
Sopraglottico: Sopra la glottide.

Per esempio, la trachea e i polmoni sono strutture subglottiche, mentre la faringe è una struttura sopraglottica.



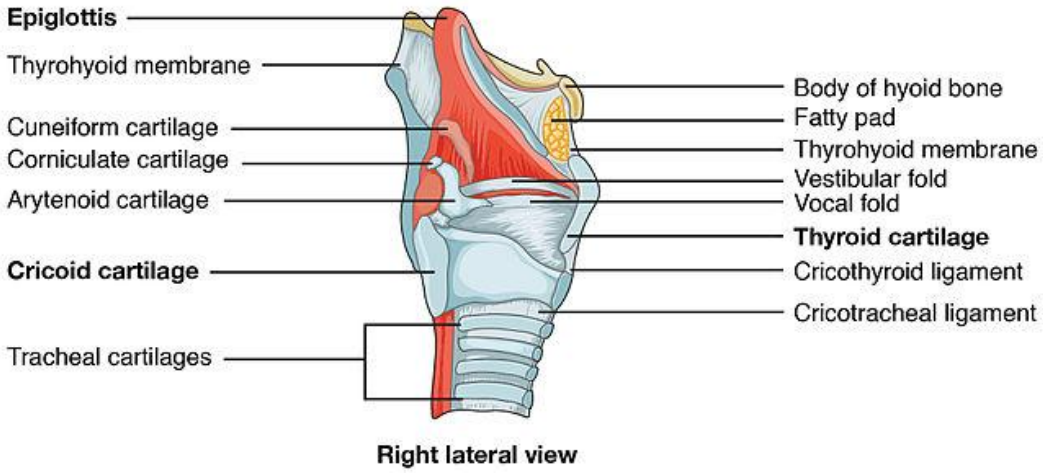


Figura 3: Laringe

La figura 4 è un corte trasversale in cui si possono vedere le corde vocali e il suo aspetto.

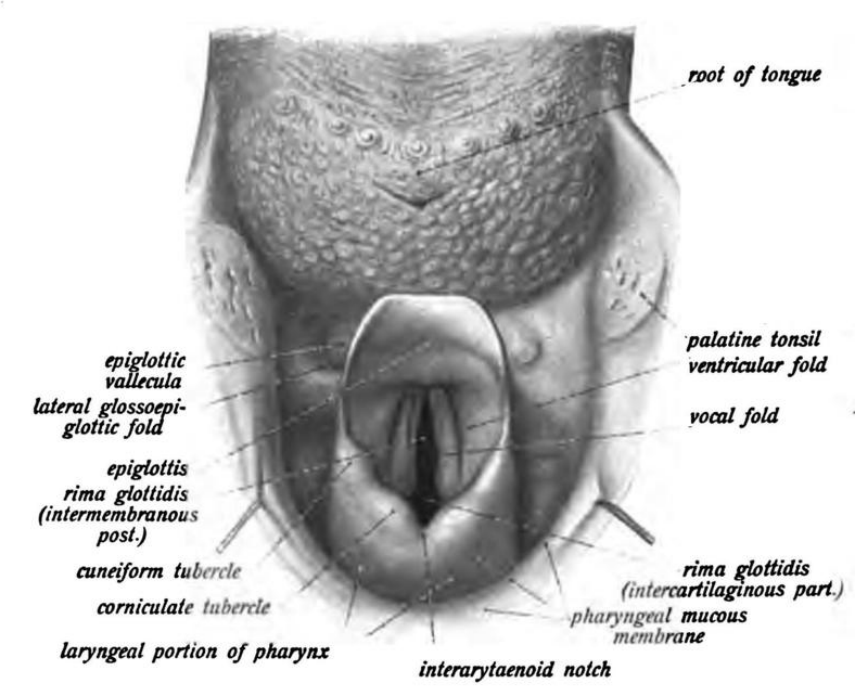


Figura 4: Corde vocali

Assieme alla figura 5, in cui si ha uno schema delle diverse posizioni che può tomare secondo se si sta parlando oppure no, mostrano la sua posizione e il suo funzionamento. Come si può vedere in quest'ultima immagine, quando si sta respirando le corde vocali sono aperte per permettere il passo dell'aria senza ostacoli. Invece, quando si parla, si chiudono per opporre più resistenza o meno e così produrre i diversi suoni quando le corde vocali vibrano dovuto alla resistenza al passo del'aria tra di loro. Questa

vibrazione viene trasmessa per le particelle d'aria. È questo il processo di generazione della voce.

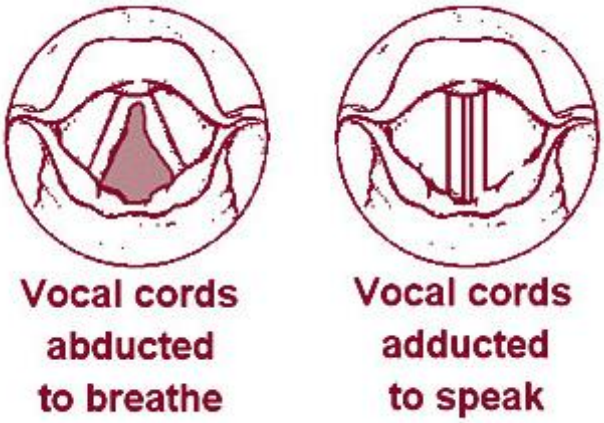


Figura 5: Schema delle corde vocali

L'aria proveniente dei polmoni passa traverso la trachea e arriva alla laringe. Una volta qui, passa attraverso le corde vocali generando questa vibrazione, questa perturbazione che è la causa del suono, e continua il suo precorso verso l'esterno per la cavità orale. È nella cavità orale dove si modula questo suono mediante una apertura maggiore oppure minore, la posizione della lingua, dei denti, delle labbra… per produrre i diversi fonemi che compongono il parlato. Per questa ragione, la segnale che mostra un microfono d'aria è più complessa di quella del microfono a contatto che non ha ancora tutta questa modulazione.

Si ha cercato di capire meglio il funzionamento delle corde vocali con diversi modelli matematici per la importanza che ha il parlato per l'essere umano. Un esempio di questi modelli si mostra nella figura 6.

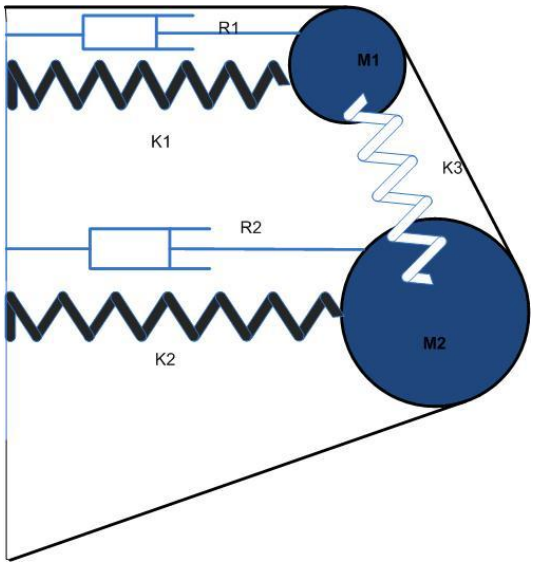


Figura 6: Modello delle corde vocali

Le vibrazione che sono la causa del suono se trasmettono per l'aria verso l'esterno permettendo così la generazione della voce. Questa vibrazione se trasmette anche per i muscoli ed è così come funziona il microfono a contatto, microfono di gola o laringofono. Il sensore di questo dispositivo si mette nella base del collo, all'inizio della laringe, sopra il muscolo cricotiroide, dove si sentono di più le vibrazione.

1.2.1 Sorgenti di suono secondarie nella laringe

Le turbolenze nel flusso d'aria della glottide sono una sorgente di suono addizionale. Ha una caratteristica sibilante che è chiamato aspirazione quando si combina con la vibrazione delle corde vocali e sussurro quando quando non ci sono queste vibrazione delle corde vocali. Ci sono anche altre suoni che vengono prodotti per subite inizi o fermate di parlato. Questi suoni di transizione sono i cosiddetti "glottal clicks" o "glottal stops".

Per ultimo, un suono secondario può essere generato per il movimento di liquidi nelle corde vocali oppure vicino loro. Una piccola quantità di questo tipo di suono è tollerabile , ma una eccessiva umidità si considera un disordine della voce. In generale, tutte queste caratteristiche si possono estrarre del spettro di frequenza.

Il proposito di tutte queste figure è quello di capire la generazione del suono, la propagazione è perchè si possono prendere campioni significativi dal esterno come si fa. Tutto questo non è possibile senza comprendere prima i principi fisici della generazione del suono, la fisiologia e funzionamento degli organi del corpo che riguardano.

1.3 Caratteristiche della voce

I suoni vocali possono essere classificati in tre classi diverse d'accordo con la eccitazione per la cui si producono. I suoni parlati sono prodotti forzando il passo del flusso d'aria attraverso la glottide, con la tensione delle corde vocali, queste vibrano e producono pulsi quasi periodici che eccitano il tratto vocale umano.

I suoni fricativi o di non parlato, si generano per la formazione di costrizioni in diversi punti del tratto vocale. Quando l'aria viene forzato a passare per questi punti ad una velocità abbastanza alta, si producono turbolenze che sono la fonte di eccitazione del tratto vocale.

La maggior parte delle lingue si possono descrivere come una serie di suoni distintivi, i fonemi. Ognuna ha un numero di fonemi diverse, tra 11 e 141. Per esempio l'inglese ha 42 fonemi diverse tra vocali, semivocali, consonanti e dittonghi, mentre l'italiano ne ha 31 e lo spagnolo 22 tra vocali e consonanti.

1.3.1 Vocali

I suoni vocali si producono per la eccitazione di un tratto vocale fisso per pulsi d'aria quasi periodici prodotti per le vibrazioni delle corde vocali. Nel caso dei suoni dei dittonghi la eccitazione è la stessa, l'unica differenza è che il suono si produce variando soavemente il tratto vocale da la configurazione di un vocale all'altra.

1.3.2 Consonanti

Semivocali:

Si chiamano così perché hanno una natura similare alle vocali. Si caratterizzano soprattuto per una transizione scorrevole tra le configurazioni del tratto vocale di due fonemi adiacenti.

Nasali:
Sono prodotti per una eccitazione della glottide con il tratto vocale totalmente costretto in diversi punti. Il flusso d'aria è forzato a passare per il tratto nasale. La cavità orale resta come camera di risonanza per certe frequenze naturali.

Unvoiced fricatives:
Si producono per una eccitazione del tratto vocale per un flusso d'aria costante che diventa turbolento nella regione di costrizione. La localizzazione di questa costrizione è quello che fa la differenza tra i suoni delle diverse consonanti di questo tipo.

Voiced fricatives:
Sono gli omologhi dei unvoiced fricatives . La differenza tra di loro è che nella generazione di questi sono coinvolti due sorgenti. Le corde vocali vibrano e quindi una delle sorgenti di eccitazione si trova nella glottide, mentre l'altra è la costrizione che fa diventare il flusso d'aria turbolento vicino a questo punto.

Voiced stops:
Queste consonanti sono eventuali, suoni non continuativi prodotti per un aumento di pressione, tra una costrizione totale in un certo punto del tratto vocale, e un rilascio di questa pressione subito dopo. Come prima, anche in questo caso la localizzazione della costrizione è la differenza tra i diversi suoni.

Unvoiced stops:
Sono simili alle sue omologhi. La differenza tra di loro è che durante il periodo di chiusura totale per la costrizione, mentre la pressione aumenta, le corde vocali no vibrano. Quindi, dopo questo periodo, quando la pressione viene rilasciata, c'è un breve intervallo di frizione, seguito per un periodo di aspirazione prima che cominci la eccitazione udibile.

Per quanto riguarda a questo progetto, i suoni verrano divisi in quelli generati per la vibrazione delle corde vocali e quelli generati per la turbolenza del flusso d'aria dovuta ad una costrizione del tratto vocale, senza entrare in tutte queste caratteristiche diverse. Lo importante è il luogo dove viene generato il suono, la sorgente di eccitazione perché indica quali sono registrabili per il microfono d'aria e il microfono a contatto e quali solo per il microfono d'aria.

I tratti vocale e nasale si possono vedere come tubi di area trasversale non uniforme. Il suono generato come si ha spiegato prima, si propaga giù per questi tubi e lo spettro di frequenza si define per il tubo. Quando una onda viene attraversa un tubo, si produce un effetto simile agli effetti di risonanza che accadono negli instrumenti musicali d'aria.

1.3.3 Risonanza
La risonanza è un fenomeno che consiste nella amplificazione di un suono per un sistema acustico dove la frequenza del suono corrisponde con una delle frequenze naturali di vibrazione del sistema. Se si parla di risonanza acustica semplicemente si riducono queste frequenze al rango di audizione umano.

Normalmente un oggetto risonante ha più di una frequenza di risonanza, specialmente nelle armoniche della risonanza o frequenza principale o fondamentale

(Frequenza più bassa dello spettro di frequenze). Vibrerà più facilmente a queste risonanze e meno alle altre.

In questo caso, il sistema risonante sarebbe il tubo d'aria se si parla degli instrumenti musicali e il tratto vocale se si parla della voce. La risonanza in un tubo d'aria sta relazionata con le sue dimensioni, forma e dipende anche da se è un tubo aperto oppure chiuso alla fine. Nel caso dei tubi degli instrumenti musicali che servono come modello in questo caso, hanno forma cilindrica.

Le colonne d'aria vibrante hanno anche risonanza negli armonici (Si chiamano armonici alle frequenze che sono un multiplo entero della frequenza fondamentale) e le frequenze di risonanza seguono le seguenti formule:

Per i tubi aperti:

$$F = n \cdot v \cdot 2L \qquad F = n \cdot v \cdot 2(L + 0{,}8d)$$

Per i tubi chiusi:

$$F = n \cdot v \cdot 4L \qquad F = n \cdot v \cdot 4(L + 0{,}4d)$$

dove n è un numero entero che rappresenta il nodo di risonanza, L è la longitudine del tubo, v è la velocità del suono nell'aria e d è il diametro del condotto. Con la prima formula per ogni caso, si ottengono le frequenze approssimative, la seconda è una formula più accurata.

Partendo da queste formule; conoscendo la relazione tra la frequenza, la longitudine d'onda e la velocità e con n=1; si può ottenere facilmente il valore di detta longitudine d'onda:

Per i tubi aperti:	Per i tubi chiusi:
$\lambda = 2(L + 0{,}8d)$	$\lambda = 4(L + 0{,}4d)$

Gli armonici dipendono della forma e delle dimensioni del tratto vocale nello stesso modo che accade con gli instrumenti musicali. Ogni forma si caratteriza per una serie di armonici. I diversi suoni si producono cambiando la forma del tratto vocale come già riferito. Quindi, le propietà dello spettro del segnale vocale, variano con il tempo quando la forma del tratto vocale cambia. Anche se per ogni laringe ci sono le sue caratteristiche per la longitudine di questa, le persone hanno certo controllo di detta longitudine e pertanto di la frequenza del suono mediante il movimento su o giù della laringe.

Il tratto vocale attua come un megafono, ingrandendo il suono prodotto per la glottide e radiato per la bocca. Anche se non c'è una potenza di amplificazione (Il tratto vocale è un sistema passivo, solo dissipa energia), certe frequenze sono favorite rispetto ad altre. In particolare, gli armonici vicini alla frequenza fondamentale sono favorite, mentre gli armonici tra frequenze naturali sono attenuate. Questa differenza si può osservare nel spettro di frequenze delle due segnali e anche nella onda generata e registrata per ogni microfono.

1.3.4 Acustica

L'acustica è definita per la ANSI/ASA come: "(a) Science of sound, including its production, transmission, and effects, including biological and psychological effects. (b) Those qualities of a room that, together, determine its character with respect to auditory effects."[15] Cioè: La cienza del suono, incluso la generazione, propagazione e gli effetti inclusi queli biologici e psicologici oppure come le qualità di una stanza che determinano le sue caratteristiche riguardo agli effetti per l'auditorio.

Gli studi di acustica ruotano intorno alla generazione, propagazione e ricezione di onde meccaniche e vibrazioni.

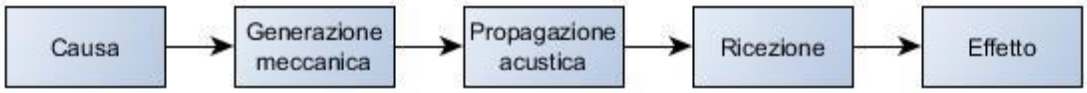


Figura 7: Percorso del suono

I passi mostrati nel diagramma della figura 7 possono essere trovati in qualunque evento o processo. Nel caso che riguarda a questo progetto interessano solo i passi mostrati nella figura 8. La generazione sarebbe la generazione della voce come si è già spiegata prima, la propagazione dell'onda acustica per l'aria e i tessuti umani (muscoli, pelle…), e la ricezione e la trasduzione per i microfoni utilizzati.

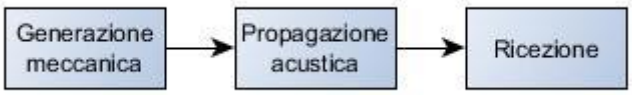


Figura 8: Percorso del suono che ci riguarda

1.3.5 Descrizione matematica delle onde

In fisica, una onda consiste nella propagazione di una perturbazione di qualche propietà (densità, pressione….) di un medio attraverso detto medio e con il coinvolgimento di un trasporto di energia senza trasporto di materia. Il medio perturbato può essere di diverse nature come aire, acqua, solido…

Gli elementi di una onda sono:

- Cresta: Punto di ampiezza massima.
- Periodo: Tempo che bisogna la onda per spostarsi di un punto di ampiezza massima al seguente.
- Ampiezza: Differenza tra il valore della cresta e il valore della posizione di riposo o equilibrio.
- Frequenza: Numero di volte che si ripete la vibrazione nella unità di tempo. È inversa al periodo.
- Longitudine di onda: Distanza tra lo stesso punto di due ondulature consecutivi, come la distanza tra due creste consecutivi.

Le onde periodiche come il suono, si caratterizzano per questi elementi, possono essere longitudinali o trasversali. Il suono è una onda longitudinale, cioè la vibrazione ha la stessa direzione della propagazione della onda. Dal punto di vista matematico, la onda più semplice è la onda senoidale che può essere descritta per la funzione:

$$f(x,t) = A \cdot \sin(wt - kx)$$

Dove A è l'ampiezza massima, w è la frequenza angolare che ha unità di radianti al secondo (w=2f), t è il tempo, x è la posizione dove del punto dove si vuole sapere l'ampiezza e k il cosiddetto numero di onda angolare che sta relazionato con la longitudine di onda (k=2).

1.4 Il suono

In fisica, si chiama così a qualunque fenomeno che coinvolge la propagazione in forma di onde elastiche attraverso un fluido e che genera un movimento vibratorio di un corpo. Il suono udibile per l'essere umano consiste in onde sonore che si producono quando oscillazioni nella pressione dell'aria diventano onde meccaniche nell'orecchio umano e sono percepite per il cervello.

Come già riferito, la voce umana si produce per la vibrazione delle corde vocali. Questa vibrazione genera una onda sonora che è la combinazione di diverse frequenze con i suoi rispettivi armonici. ogni segmento del suono del parlato viene caratterizzato per un certo spettro di frequenze. L'orecchio umano è capace d'identificare diversi formanti di detto suono e percepire ogni suono che ha diverse formanti come qualitativamente diversi e distinguere, per esempio, diverse vocale tra di loro.

Come si ha spiegato prima, la frequenza fondamentale dipende delle caratteristiche del condotto vocale, delle dimensioni della laringe. È questa la ragione delle differenze tra le voci maschili, femminili e dei bambini. La voce maschile ha la sua frequenza fondamentale o naturale tra 100 e 200 Hz mentre la voce femminile è più acuta e ha la sua frequenza naturale tra 150 e 300. I bambini hanno la voce ancora più acuta. Senza il filtrato per risonanza che produce la cavità buco-nasale, le emissioni sonori non sarebbero abbastanza chiare per esser udibile. E questo processo quello che permette generare i diversi suoni dei fonemi.

La voce umana è una onda di ampiezza massima variabile, formata per la frequenza fondamentale e le sue armonici. Come si è visto prima, matematicamente una onda viene definita per la sua frequenza e ampiezza. Per questa ragione i parametri misurati nel progetto sono la F0 o frequenza fondamentale e la Vrms. Il secondo parametro non è l'ampiezza per la variabilità di questa nel parlato e per gli armonici. Un parametro più utile in questi casi per avere una idea dell'energia della onda qualsiasi sia la sua forma è il Vrms (Valore efficace o valore quadratico medio) che ha la seguente espressione:

$$V_{rms} = \lim_{T\to\infty} \sqrt{\frac{1}{T}\int_0^T [f(t)]^2 dt}$$

Nel caso di onde senoidali pure si può semplificare a:

$$V_{rms} = \frac{V_p}{\sqrt{2}}$$

Con queste due parametri, la voce è caratterizzata.

L'altro obiettivo del lavoro sviluppato è quello di prendere dell'informazione per esseri capaci in un futuro di trovare una relazione tra la variazione di questi parametri e il rischio di certe disturbi nelle corde vocali. Si spiegarà più avanti perché è possibile raggiungere questo secondo obiettivo con questi parametri.

2. STATO DELL'ARTE

In questo capitolo, si fa un resumen dei diversi sistemi di monitoraggio della voce esistenti attualmente.

Si possono trovare diversi dispositivi sviluppati per il monitoraggio della voce con l'obiettivo di essere capaci di individuare un rischio per la salute del sistema vocale. I lavori di ricerca documentati più recenti in quest'area sono stati sviluppati per il Centro Nazionale della voce e il parlato (National Center for Voice and speech: NCVS) e per l'Ospedale Generale di Massachusetts. Si ha sviluppato anche un dispositivo chiamato VoxLog per la Università di Linköpings in Svezia per questo proposito.

I tre dispositivi sono portabili e servono ad analizzare basati nel misuramento del livello di accelerazione della pelle (Skin acceleration level: SAL) dovuto alla vibrazione delle corde vocali. A continuazione si farà una breve descrizione di questi dispositivi. I suoi obiettivi e come lavorano.

2.1 NCVS

Si tratta di un dispositivo sviluppato per un centro di ricerche nel 2004 con l'obiettivo di monitorare la voce di professionali come profesori, cantanti…

Un uso eccessivo della voce può condurre a problemi relativamente frequenti della voce. Per questa ragione, c'è la necessità di misurare il tempo di utilizzo della voce così come le sue caratteristiche per questi professionali della voce e monitorare anche la ricorrenza di questi problemi per stabilire limiti adeguati.

Prima di questi dispositivi, si avevano sviluppato altre che misuravano la frequenza fondamentale e il voicing time oppure l'intensità e il voicing time per periodi di 12 ore ma nessuno di questi dispositivi era possibile commercializzarlo.

L'obiettivo di questo dispositivo è quello di riconoscere i segmenti di parlato di un discorso preso da un singolo microfono portato per il soggetto; calcolare l'intensità (misurata come livello di pressione sonora o SPL) e la frequenza fondamentale dal segnale del microfono e fare un registro temporale dei parametri nel corso della giornata.

2.1.1 Parametri misurati

Il primo passo è quello di specificare i parametri misurati per quantificare il carico vocale. I disordini delle corde vocali causati per la intensità del parlato oppure per lunghi tempo di parlato, si possono pensare come problemi di esposizione e l'esposizione del tempo si misura per quello che in inglese si chiama "dose", cioè, la quantità di utilizzo nel tempo.

"Time dose" è una misura del tempo totale (in secondi) durante il quale stano vibrando le corde vocali, è un indicatore di quando le corde stanno veramente vibrando.

Quindi, è chiaro che per calcolare l'esposizione della voce è necessario avere un registro temporale di questi parametri: kv, SPL e F0 per calcolare il dosado con le formule indicate in questo articolo.

2.1.2 Disegno

Il dosimetro poteva essere implementato in un dispositivo fatto a proposito o come dispositivo de software virtuale in un computer personale. Per i propositi del progetto si sceglie un Pocket PC per i vantaggi che proporciona: Processore potente dedicato, gran capacità di memoria e un sistema di 32 bits basato su Windows, cioè, familiare. Inoltre, più si abbassava il prezzo dei Pocket PC più competitivo diventava il dispositivo e così diminuivano i rischi dell'osolescencia programmata.

Il disegno del dosimetro si riduce quindi allo sviluppo del software per il Pocket PC e riguardo al hardware, si deve soltanto scegliere il microfono adeguato da collegare al PC e la forma migliore di attaccare il microfono alla pelle.

2.1.3 Descrizione del programma

Il programma sta disegnato per calcolare il SPL, la F0 e il tempo di parlato per metodi semplici e anche per salvare i dati di forma efficiente. Questo dispositivo aveva anche la posibilità di controllare ogni due ore il funzionamento e salvataggio dei dati.

Calcolo dei parametri

Per poter fare l'analisi dei dati, si devono dividere in finestre o frames. Si sceglie un frame di 30 ms per poter dettetare/individuare le pause intersilabiche del parlato tipico. I campioni si normalizano tra (-1, 1)V. Con questi dati, si calcola il Vrms di ogni frame, il cui serve a ottenere il SPL di detto frame. Per sapere la relazione tra Vrms e SPL, fano prima la calibrazione.

Per calcolare il tempo reale di parlato, si mette una soglia di forma che tutti quei dati che restano sotto si considerano di silenzio o di rumore di fondo e quelli che superano questa soglia si considerano di parlato, così si può ottenere la percentuale di parlato di ogni frame facilmente.

Per ultimo, resta il calcolo della F0, la frequenza naturale o fundamentale. L'analisi come si ha detto prima si fa per ogni frame, e il metodo scelto è stato l'algoritmo FFT per la sua semplicità e afidibilità, facendo uso dopo di un altro algoritmo per correggere i problemi dovuti alle tracce di subarmonici comunque. Alla fine di tutto questo processo i test fatti confermano la acurabilità dei risultati ottenuti dal processo.

2.2 Portable Vocal Accumulator (PVA), Ospedale di Massachusetts:

In questo caso si parla di un dispositivo sviluppato per l'ospedale di Massachusetts nel 2003. I loro obiettivi erano sviluppare un nuovo dispositivo per il salvataggio di dati sulla voce di individui durante tutto il giorno e testare il dispositivo con gente con disordini vocali e gente senza questi problemi.

Questo dispositivo dovrebbe permettere stabilire un regolamento, delle norme per l'uso della voce per i diversi cittadini, per esempio nelle diverse settori lavorali; l'investigazione del ruolo del uso della voce nei disordine vocali; Il monitoraggio di informazione oggettiva del uso della voce giorno dopo giorno di persone con disordini vocali. Inoltre di permettere alle persone avere un feedback in tempo reale del suo utilizzo della voce.

Utilizza un accelerometro per misurare le vibrazioni della pelle del collo causate durante il processo di fonazione. Il segnale del accelerometro è analizzato in tempo reale per stimare la frequenza fondamentale, il livello di pressione sonora e la periodicità. Questi parametri si salvano nel PVA e possono essere scaricati in un computer personale collegando il dispositivo mediante un porto serie.

Fino a quel momento, i dispositivi sviluppati a cui fa riferimento, facevano monitoraggio o feedback ma non facevano le due funzioni e quasi tutti erano troppo grandi per essere portabili.

2.2.1 Metodo di disegno del PVA:

Il dispositivo doveva:

- Interferire il meno possibile la normale attività vocale.
- Misurare di forma simultanea diversi parametri relevanti come la F0, il SPL e la durata.
- Avere un funzionamento continuo nel prendere i dati al meno per 12 ore.
- Lasciare aperta la possibilità di un maggiore sviluppo posteriore come per esempio poter misurare altre parametri.

2.2.2 Hardware:

Il PVA fu disegnato intorno a un processore di segnali digitali (DSP) con delle caratteristiche tale che aveva dimensioni piccole (12 x 8.5 x 2 cm) ed era leggero. Il suo consumo di energia era abbastanza basso di essere capace di funzionare di forma continua per 12 ore. L'uso di questo processore poteva anche essere programmato per e offrire un feedback con un led e un piccolo vibratore.

Invece di utilizzare microfoni come i dispositivi sviluppati prima, usa un piccolo accelerometro messo sulla pelle all'esterno della mela di adamo che prende il segnale della voce. L'uso di un accelerometro al posto di un microfono porta una serie di vantaggi per questo tipo di applicazioni. Per primo, le piccole dimensioni e la posizione dove si mette lo fa disturbare molto di meno rispetto ai microfoni d'aria utlizzati prima e al microfono di laringe (meter referencia aqui). Per secondo, l'accelerometro e quasi immune al rumore di fondo e per terzo e ultimo, prende soltanto segnale del parlato, della voce. Contrariamente i microfoni d'aria prendono segnale anche dei rumori di fondo e di suoni che non corrispondono alle vibrazioni delle corde vocali. Tutta questa informazione in più faceva diventare il segnale troppo complesso per l'analisi desiderato. Come si ha spiegato prima il segnale presso per un accelerometro o microfono a contatto non ha tutta la modulazione che introduce la cavità orale e per questa ragione e molto più semplice.

2.2.3 Software

Il software del PVA fu sviluppato prima in un PC e dopo nel prototipo del PVA. Ogni 125 ms applica l'algoritmo. Prima di tutto divide il frame in piccole parti di 25 ms e calcola il valore del Vrms per ogni di questi sottointervalli. (idea para trabajo/mejora futura). Se il valore di Vrms calcolato e superiore a una soglia impostata previamente, vuol dire che si tratta di un frame di parlato e si calcola la F0 utilizzando metodo di autocorrelazione.

Per ogni frame, il PVA salva tre dati: Il tempo in cui inizia rispetto all'inizio totale, il livello (Il valore di questa variabile si salva come negativo se il frame non è periodico) e il valore della F0.

La durata del parlato puó essere calcolata comparando il numero di frames marcati come frames di parlato con quelli di non parlato in una durata totale di 125 ms. I dati sono salvati in formato di testo nel computer e anche nel proprio dispositivo.

2.2.4 Test del PVA

Durante lo sviluppo del dispositivo, si fanno diversi test per validare le sue diverse funcionalità e anche come ricerca delle possibilità che offre tanto per il software come per il hardware.

2.3 Inter-noise 2009

In questo caso si parla di un dispositivo sviluppato per l'università di Gothenburg in Svezia. I dosimetri di rumore registrano rumore esterno e anche il suono della voce propria come se fosse rumore. Diversi studi su uccelli e mammiferi rivela che e il rischio di perdita dell'udito per l'esposizione alla propria voce e molto basso.

Quindi, può essere uno dei interesi per realizzare degli studi sul rumore con un dosimetro che scluda la voce umana dell'analisi. Questa è la motivazione per la sua ricerca nel dettetare la voce attiva. Come è stato visto negli studi previ, è molto più accurato un microfono di gola formato per un accelerometro che uno d'aria. Per questa ragione l'obiettivo di questo grupo di ricerca è quello di evaluare come un accelerometro attaccato alla pelle soltanto per pressione, senza colla, reagisce al rumore dell'aria e a quelli originati per il portatore dell'accelerometro.

Lo studio svolto consiste nel collegare un microfono d'aria e il accelerometro o microfono a contatto, ad un registratore di segnali digitale e fare e seguenti test su diversi soggetti:

- Silenzio: Ne parlato ne rumore di fondo.
- Rumore: Soltanto rumore di fondo, senza parlato
- Parlato debole: lettura di un testo a bassa voce.
- Parlato forte: lettura di un testo ad alta voce.
- Movimento: Correre nella stanza avanti e indietro senza parlare.

Nei risultati si vede che il segnale durante il parlato debole è notevolmente più signficativo di quello presso durante i test di rumore per lo quale, si dimostra che il accelerometro può essere utilizzato per la dettezione del parlato, della voce attiva.

3. DISPOSITIVO E LAVORO SVILUPPATO:

In questo capitolo, si spiega il lavoro sviluppato, i componenti del dispositivo e come funziona. Si tratta di una nuova versione del Voice-Care, dispositivo sviluppato durante gli ultimi anni per il Dipartimento di Elettronica e Telecomunicazioni (DET) del Politecnido di Torino.

3.1 Introduzione:

Come già riferito, lo scopo di questo lavoro è quello di sviluppare un dispositivo con autonomia per tutto un giorno lavorando di forma continuata. Si tratta di un dispositivo simile a quelli di cui si ha parlato nello stato dell'arte. L'obiettivo è registrare di forma continuativa il parlato e prendere frames di campioni per calcolare i parametri scelti per analizzare le caratteristiche del parlato e dell'ambiente in cui si fa deta registrazione.

Il dispositivo sta formato per:

- Arduino due (Figura 9)
- Microfono a contatto o di gola (Figura 10)
- Microfono d'aria (Figura 12)
- Sensore di temperatura e umidità Sensirion (Figura 13)
- Amplificatori programmabili (Figura 11)

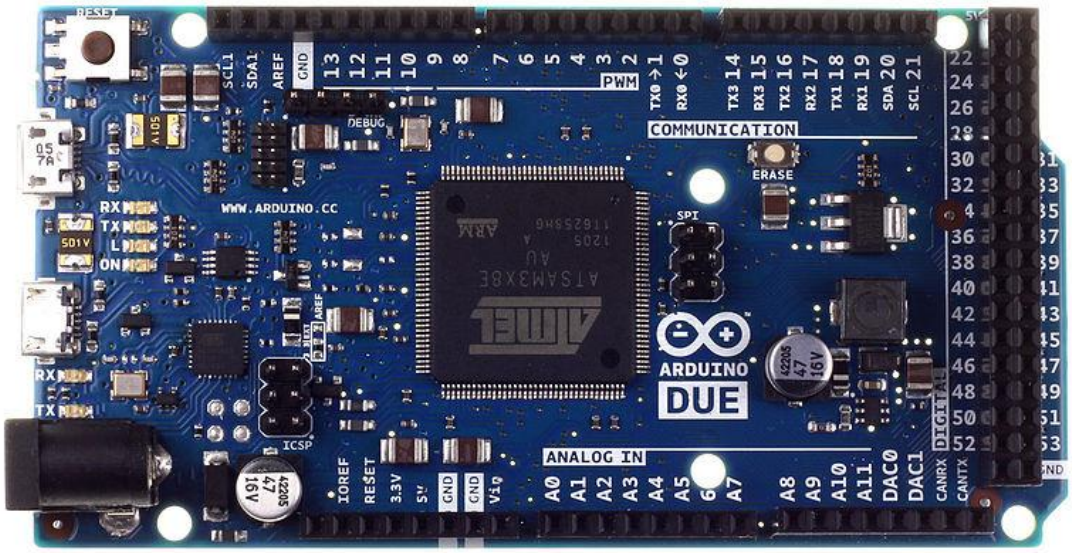


Figura 9: Arduino due

Figura 10: Microfono di gola

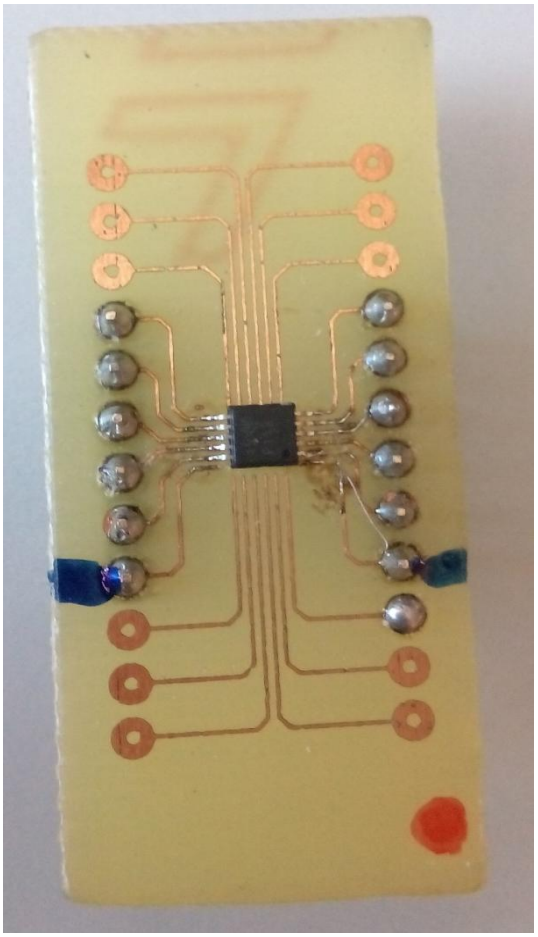


Figura 11: Amplificatore programmabile

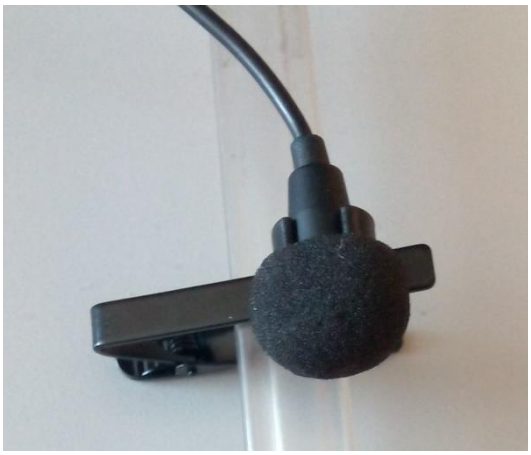


Figura 12: Microfono d'aria

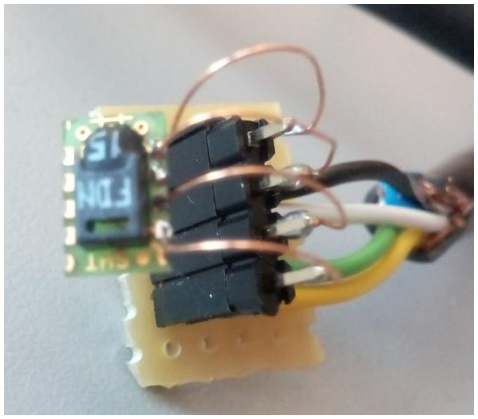


Figura 13: Sensirion

La registrazione del parlato si fa mediante un accelerometro o microfono di contatto attaccato alla gola per un nastro medico (quello che atenua di meno le vibrazioni). Con questo microfono si prende il segnale vocale prima della modulazione della cavità orale per calcolare la F0 e il Vrms e il valore del tempo di di parlato rispetto al tempo totale di registrazione e avere così il Dt calcolato per ogni frame nell'elaborazione posteriore dei dati con matlab. Così si può avere una caratterizzazione della voce e avere anche la possibilità di fare un analisi posteriore per individuare il rischio di soffrire disturbi nelle corde vocali.

Per avere una informazione più completa dell'ambiente in cui si hanno preso i dati e così poter analizzare le possibili cause di parlato più forte del normale per un soggetto per esempio, si collega anche un microfono d'aria il cui compito è quello di prendere il rumore di fondo nel modo di monitoraggio e servire alla calibrazione nel modo di calibrazione. Per questo stesso scopo si ha aggiunto anche un sensore di temperatura e umidità e avere così una caratterizzazione dell'ambiente in cui si prendono i dati.

Inoltre, il segnale preso per i microfoni è molto bassa. Per migliorare la qualità dei dati presi si ha messo una etapa di amplificazione programmabile tra i microfoni e il microprocessore utilizzato.

3.2. Descrizione generale:

3.2.1 Dispositivo iniziale:

Come si ha spiegato nell'introduzione è stato introdotto qualche miglioramento rispetto alla prima versione del dispositivo(VC) sviluppata per il dipartimento. Questa prima versione si trattava di un dispositivo portabile che acquisiva dati grezzi, campioni del segnale vocale e gli salvava in una scheda SD. I dati erano dopo processati per ottenere i parametri (Livello di pressione sonora SPL, Frequenza fondamentale F0 e Tempo di fonazione Dt) necessari per individuare e prevenire cativi comportamenti vocali e disordini causati per questi.

Il VC ha una componente hardware e una software caricata nel microcontrollore della scheda. Il dispositivo poteva funzionare in due modi a scelta, ognuno col suo scopo, calibrazione e monitoraggio.

Nella calibrazione aveva due canali di ingresso, uno per il microfono a contatto che misura il Livello di accelerazione sonora (SAL) attraverso le vibrazioni della pelle nel collo e l'altro per il microfono d'aria che misura il livello di pressione sonora (SPL) utilizzato per fare la calibrazione e poter trasformare SAL in SPL che è l'unità di misura dell'intensità sonora. In questo modo di funzionamento salva soltanto i dati di un piccolo periodo di tempo.

Nel monitoraggio lavorava soltanto con uno dei canali, quello col accelerometro collegato per fare la registrazione dei dati grezzi del segnale vocale generato. In questo modo di funzionamento invece salva dati di forma continua chiudendo un file ogni 3 minuti finchè non si spegne o ferma.

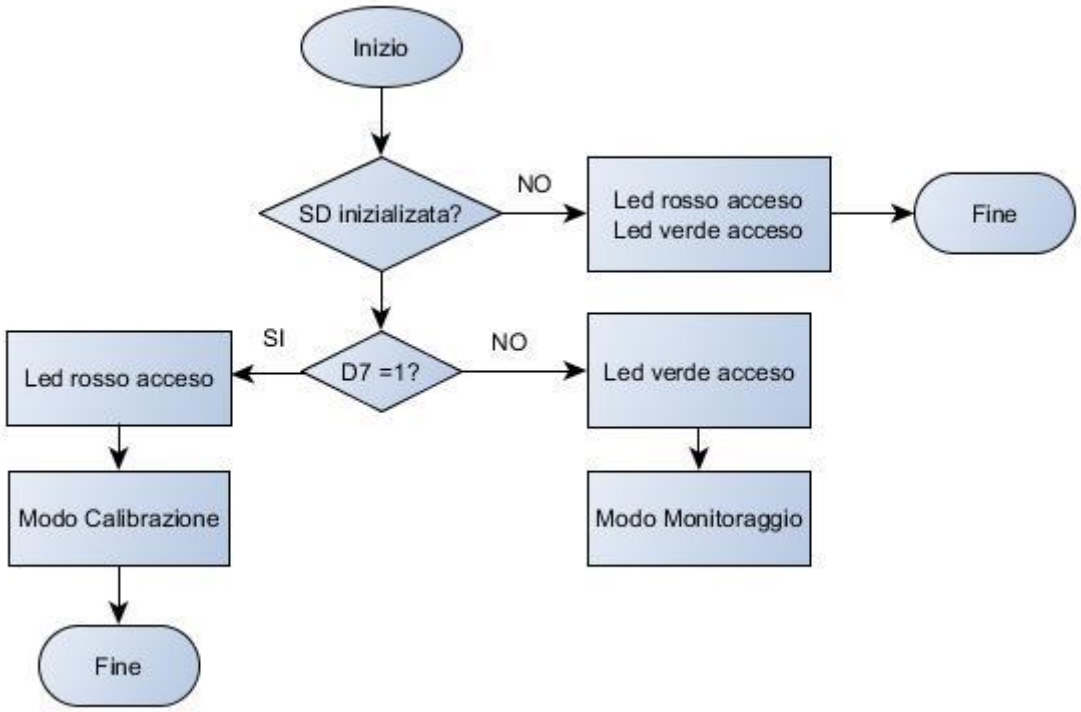


Figura 14: Diagramma di flusso primo VC

Il processo segue il diagramma di flusso mostrato nella figura 14. I campioni si prendono con una frequenza di campionamento di 38460 campioni per secondo nella calibrazione e 19230 nel monitoraggio. I campioni sono analogici per cui si prendono con il convertitore analogico-digitale (ADC) della scheda. Ogni frame ha una durata di 30 ms.

In questo primo prototipo si trovarono alcuni problemi come frames perduti dovuto al clock del microcontrollore del Arduino Uno (ATMEL ATmega328), cioè, non era veloce abbastanza per salvare tutti i dati nella SD nell'intervallo di tempo a disposizione. Come non riusciva a fare dei calcoli diventava semplicemente un dispositivo per prendere dei campioni e non tutti per le perdite commentate.

3.2.2 Primo miglioramento

In questo primo miglioramento del dispositivo si fecero alcuni cambi. I più importanti il cambio della scheda per un altro modello di Arduino più potente, l'Arduino due che ha un altro microcontrollore (ARM SAM-3X8E), e l'aumento del numero di canali di ingresso. Come nel primo prototipo, si utilizza la tenica del doble buffer, cioè, mentre si riempie uno con i campioni, si lavora sul l'altro.

Mentre la versione anteriore non riusciva a fare i calcoli nell'intervallo che riempiva di campioni uno dei buffer e neanche a salvare tutti i dati nella SD, con questo altro controllore riesce a ottenere i parametri. Grazie a questo i valori a salvare sono molti meno, soltanto un valore di ogni parametro per frame invece di tutti i valori dei campioni e così è capace di calcolare i parametri e di salvarli nella SD

Si risolve di questa forma i due problemi principali che aveva: non si perde informazione perche salva i dati di tutti i frame e non è solo un dispositivo di campionamento ma anche di analisi del segnale presso, assicurando l'integrità del funzionamento continuo. La nuova scheda ha anche dei miglioramenti nell'ADC con una risoluzione maggiore, 12 bits invece di 8.

In questa nuova versione del dispositivo, inoltre alla risoluzione dei problemi, si aggiungono nuove funzionalità.

- Si controlla meglio la frequenza di campionamento, di forma que invece di essere un valore multiplo dei valori dati per l'Arduino, si può mettere la frequenza desiderata col clock. La frequenza di campionamento nel modo di calibrazione si imposta a 30.000 campioni al secondo e quella di monitoraggio a 20.000, cioè a 30kHz e 20kHz.

- Tanto nel modo di calibrazione come nel monitoraggio, si aggiunge un altro canale di ingresso per avere informazione anche del rumore di fondo così, in questa versione si hanno tre canali nella calibrazione e due nel monitoraggio.

In questa versione si studia il metodo migliore (sempre parlando del presente caso) di fare il calcolo della frequenza fondamentale. Si studia il metodo della FFT e la correlazione e alla fine si sceglie la correlazione per la differenza di carico computazionale per il microcontrollore, per assicurare come si ha spiegato prima il campionamento continuo durante l'uso del dispositivo e la integrità dei valori salvati.

3.2.3 Ultimo miglioramento

Per ultimo si ha la versione più attuale del dispositivo che è quella svolta in questo progetto. Si continua con la stessa scheda e per tanto con lo stesso microcontrollore (ARM SAM-3X8E), ma si fanno dei cambiamenti sui canali e le funcionalità.

- Si toglie il terzo canale per il modo di calibrazione lasciando soltando due canali in entrambi modi.

- Si aggiunge il sensore di temperatura e umidità (Sensirion SHT15) per avere una caratterizzazione più completa dell'ambiente. Questo sensore si comunica con l'Arduino per due pin digitali.

- Si aggiungono anche due amplificatori programmabili, uno per ogni canale di ingresso, cioè, per ogni microfono collegato.

Tutti questi miglioramenti, e il funzionamento di tutto il dispositivo si spiega nelle prossime sezioni.

3.3 Implemetazione:

Come si ha già spiegato, si tratta di un dispositivo il cui scopo principale è quello di prendere campioni del segnale vocale, calcolare e salvare i parametri specificati di forma continua durante tutto il tempo di funzionamento. Inoltre a questo scopo principale si hanno aggiunto altre funcionalità.

Anche se l'obiettivo è quello di ottenere i valori dei parametri che servono a caratterizzare la voce e a poter sapere il rischio di disordini vocali mediante l'analisi di questi parametri, a volte bisogna vedere i segnale completo, i dati grezzi, non solo per verificare i calcoli o il funzionamento delle funzionalità del dispositivo ma per avere delle registrazioni per analisi e studi futuri magari con un altro scopo o approccio.

L'analisi del segnale vocale si fa con i campioni presi per il microfono a contatto. I campioni presi con questo microfono sono valori di SAL come già riferito ed è questa la ragione del modo di funzionamento di calibrazione, trovare la relazione tra i valori di SAL e quelli di SPL. Una volta trovata questa relazione si può dare inizio alla registrazione, calcolo e salvataggio dei parametri di forma continua nel tempo.

Quindi, il Dispositivo ha due modi di funzionamento a scelta manuale: Calibrazione e Monitoraggio. Questa scelta si fa modificando il valore di ingresso a un pin digitale (D7). Come entrambi modi hanno bisogno della connessione con la scheda SD per il salvataggio dei valori, il primo passo a realizzare e inizializzare detta scheda e verificarlo. Se la scheda non si inizializza di forma corretta si mostra un messaggio nello schermo.Una volta fatto, si sceglie tra un modo di funzionamento o l'altro.

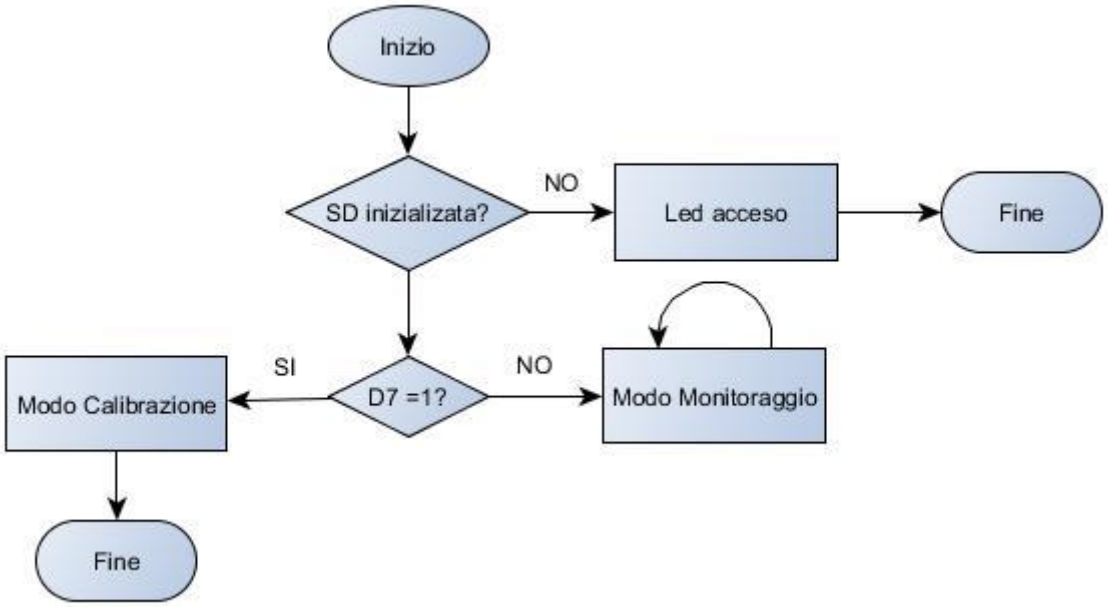


Figura 15: Diagramma di flusso generale

Una volta scelto il modo di monitoraggio, si impostano i valori dei registri della scheda necessari per il funzionamento in modo continuo. Si sceglie il temporizzatore a utilizzare, le interruzioni sono abilitate (ma solo quelle di questo temporizzatore, i resto si bloccano, non si abilitano) e si imposta un modo di funzionamento nel temporizzatore di "free running", cioè, funzionamento continuo, ogni volta che salta l'interruzione, torna di nuovo al valore iniziale. Si imposta anche di questa forma la frequenza di campionamento. Invece di impostare detta frequenza come multiplo o divissore della frequenza del clock della scheda, si imposta un valore nel temporizzatore ed ogni volta che si arriva a questo valore, salta una interruzione. Dopo, nella rutina di attenzione all'interruzione si stabiliscono i canali di campionamento, si prendono i valori e tutto quanto. La frequenza di campionamento impostata e 10000 campioni al secondo. Si abilitano anche i canali di ingresso per i microfoni, bloccando il resto, e si avvia l'ADC.

Messi apposto tutti i registri necessari per il suo corretto funzionamento si deve stabilire la rutina di attenzione all'interruzione. È in questa rutina dove si implementa la tenica del doble buffer. All'inizio del programma si hanno dichiarato due buffer di 300 dati per ogni canale (30 ms campionati a 10000 campioni per secondo). Ogni volta che salta l'interruzione, si prende un campione di un canale dell'ADC; si blocca questo canale, si abilita il secondo e si prende un campione dell'altro canale, bloccandolo di nuovo e abilitando il primo di modo che resta pronto in attesa di una nuova interruzione. L'indice di posizione dei buffer si incrementa in ogni interruzione e quando si riempie il primo buffer, si inizia a riempire il secondo (realmente due buffer perchè sono due canali). Nel tempo che riempie un buffer il dispositivo deve essere capace di fare tutti i calcoli e di salvare i dati per assicurare il funzionamento continuo.

3.3.1 SPI

La comunicazione con la scheda SD e con gli amplificatori di guadagno programabile si fa con la interfaccia seriale periferica (SPI). È un protocollo seriale sincrono di dati usato per microcontrollori per comunicare con periferici di forma veloce e a distanza corta. I componenti sono:

- Dispositivo master: In questo caso e generalmente è il microcontrollore.
- MISO (Master In Slave Out): Invio di dati del periferico al microcontrollore.
- MOSI (Master Out Slave In): Invio di dati del microcontrollore al periferico.
- SCL (Clock seriale): Clock per sincronizzare la comunicazione.
- SS (Slave Select): Si riferisce al pin che abilita la comunicazione SPI con ogni periferico. In altre modelli di Arduino si può collegare soltanto uno con questa interfaccia, ma nel caso del due si possono collegare 3. Si utilizza uno per la SD e le altre due per i due amplificatori programmabili. Nel caso di avere più di un periferico, condividono MISO, MOSI e CLK, tutto trane il SS.

Si deve quindi avviare la interfaccia SPI per ogni periferico con i diversi pin di SS. Si utilizza il pin 4 per la SD e il 10 e il 52 per gli amplificatori. Il 52 per il canale del microfono di gola e il 10 per il canale del microfono d'aria.

3.3.2 Finestra di pesatura

La finestra di pesatura è un vettore delle stesse dimensioni dei buffer riempito con la funzione:

$$w[i] = 0{,}54 - 0{,}46 \cdot \cos\left(\frac{2 \cdot \pi \cdot i}{length - 1}\right)$$

dove i fa riferimento ad ogni posizione del vettore e length alla dimensione del vettore, cioè, 300 in questo caso.

La funzione di questa finestra di pesatura è quella di attenuare gli errori causati per:

- avere un segnale discreto dovuto al campionamento
- il frame di dati scelto per l'analisi non è un multiplo esatto del periodo del segnale.

Messi apposti tutti i valori iniziali e la rutina di attenzione all'interruzione, si può iniziare col programma e i suoi modi.

3.3.3 Programma dei parametri

Nel programma di dati elaborati inoltre al campionamento dei dati si calcolano i parametri salvando solo questi dati.

Calibrazione

La calibrazione come è stato già commentato, ha la sua ragione di essere nella necessità di trovare una relazione tra i valori di SAL presi per l'accelerometro o microfono a contatto ed i valori di SPL presi per il microfono d'aria che sono le unità di misura utilizzati per il suono.

Per trovare questa relazione, si dovrà calcolare prima la costante del microfono d'aria per tradurre i valori di Vrms calcolati con i campioni presi di questo canale e i DB (unità in cui si misurano i SPL).

$$SPL = 20log\left(\frac{P}{P_0}\right) \quad \text{dove} \quad P_0 = 2 \cdot 10^5 \qquad \text{e} \qquad P = K_{\,mic} \cdot Vrms_{mic}$$

Ogni volta che si ha il buffer pieno si fanno i calcoli per ottenere il valore di Vrms e fare il controllo del guadagno.

Controllo di guadagno

Il segnale generato per il parlato normale dell'essere umano, cioè, senza gridare o susurrare, e abbastanza piccolo rispetto al fondo di scala disponibile. Anche se i parametri necessari si possono calcolare ugualmente senza questo paso, più sensibilità si ha nel campionamento, più accurati saranno i valori dei parametri e più informazione si potrà ottenere dell'analisi.

L'amplificatore utilizzato è il PGA112. Si tratta di un amplificatore programmabile con comunicazione SPI con la scheda Arduino. Questo modello ha 8 livelli di guadagno binari, cioè, per ogni livello di guadagno in più, il guadagno reale si incrementa al doppio.

Tabella 1: Tabella dei guadagni

Guadagno binario	Guadagno reale
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128

Ha diversi modi di funzionamento in funzione delle funcionalità desiderate. Per il progetto si ha scelto di lavorare con un solo canale di ingresso e una Vref= 1,65V. Si lavora solo con un canale di ingresso anche se si ha la possibilità di collegare due canali perche la differenza tra i due segnali (microfono di contatto e microfono d'aria) fa necessario un controllo differenziato del guadagno se si vuole avere la massima sensibiltà per ognuno. La ragione di aggiungere una componente di continua al segnale della metà del valore del fondo di scala dell'ADC è quella di trasformare il segnale bipolare dei microfoni in segnali unipolari per la caratteristica dell?ADC che non permette l'ingresso di valori negativi. Di questa forma, il segnale centratto in 0 originalmente, sale fino 1,65 dove è centrato adesso.

La logica di funzionamento pensata per l'algoritmo si mostra nel seguente diagramma di flusso:

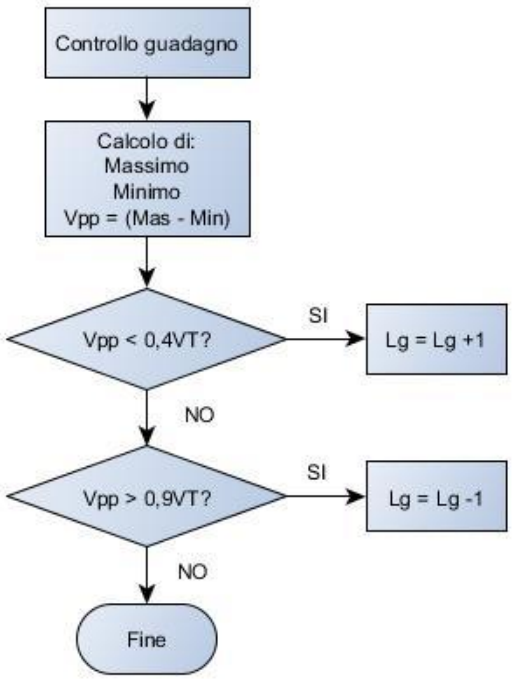


Figura 16: Diagramma de flusso: Controllo di guadagno

Cioè, si calcolano i valori massimi e minimi e la differenza tra di loro, del vettore di campioni grezzi e si confrontano questi valori con il fondo di scala trovando tre situazioni diverse:

- Se la differenza tra di loro è minore del 40% del fondo di scala, si deve aumentare il livello di guadagno.
- Se la differenza è maggiore del 90% del fondo di scala, si abbassa il livello di guadagno.
- Se la differenza si trova tra il 40% e il 90%, non si fa nessun cambiamento.

Per aumentare la sicurezza di evitare la saturazione, tanto superiore come inferiore, si aggiungono dei limiti alle situazioni anteriori e per evitare problemi di arrivare per software a valori di guadagno inesistenti, si aggiungono anche delle condizioni per non salire sopra il livello 7 ne abbassarsi sotto il livello 0.

Come si può vedere più avanti, nell'esperimento 6, quando si prova l'algoritmo con segnale reale della voce, satura troppo por lo che si devono scartare troppi frames del canale del microfono a contatto. Per questa ragione si aggiunge un ritardo nella salita del guadagno del microfono di gola, per il microfono d'aria non c'è bisogno di fare il cambiamento. Dopo i miglioramenti, le situazioni e le condizioni (per il canale senza ritardo) sono:

- Se la differenza tra di loro è minore del 40% del fondo di scala, il valore minimo di campionamento e minore di 1200 e il valore di guadagno attuale e minore di 7, si deve aumentare.

- Se la differenza è maggiore del 90% del fondo di scala, il valore minimo di campionamento e maggiore di 100 e il valore di guadagno attuale e maggiore di 0 si abbassa il livello di guadagno.
- Se la differenza si trova tra il 40% e il 90%, non si fa nessun cambiamento.

L'algoritmo finale per il canale del microfono a contatto che ha il ritardo nella salita del guadagno si mostra nella seguente figura:

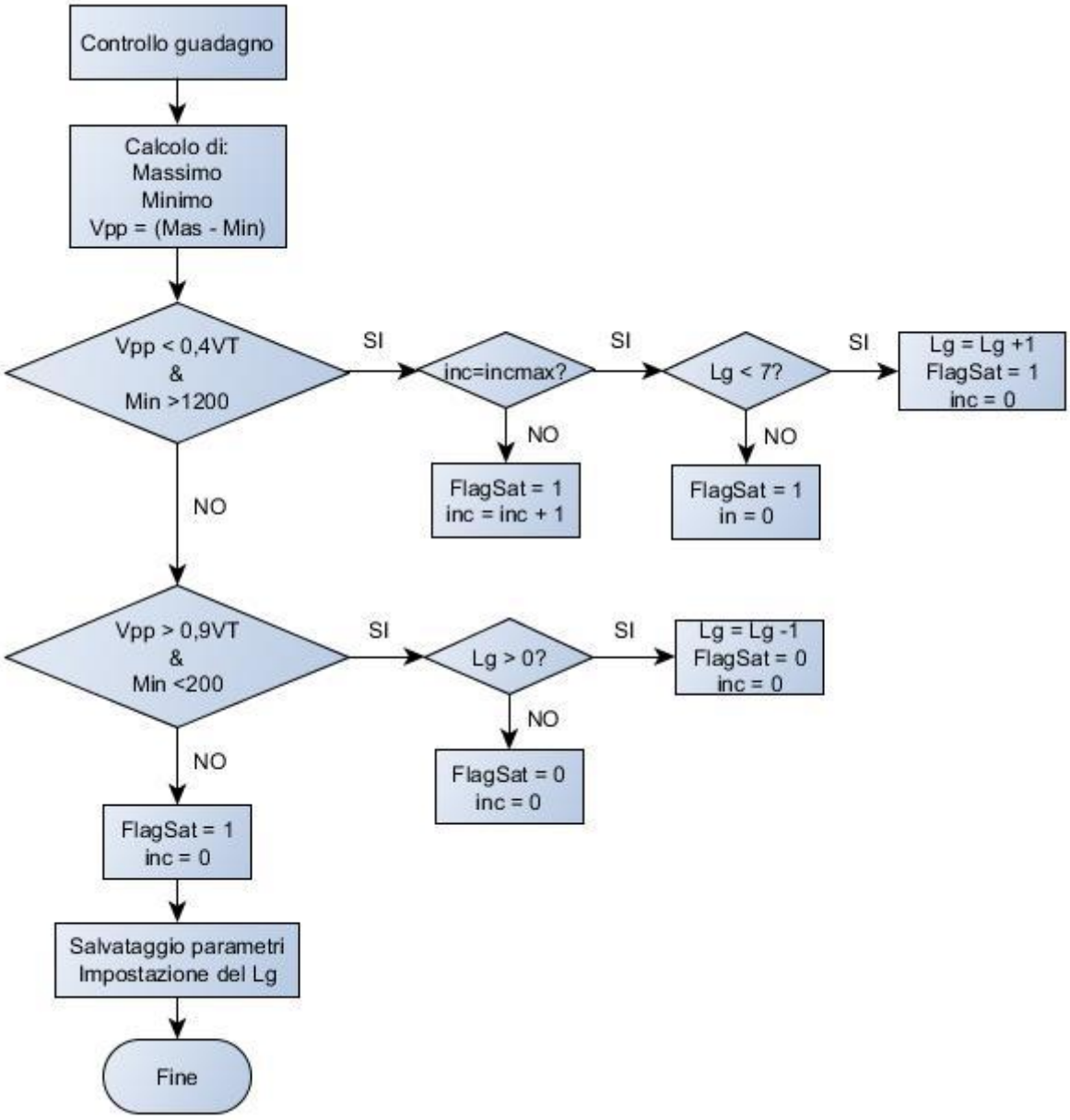


Figura 17: Controllo guadagno con incmax

Il valore del parametro incmax si può cambiare a convenienza.

L’algoritmo finale per il canale del microfono d’aria, senza ritardo nella salita è:

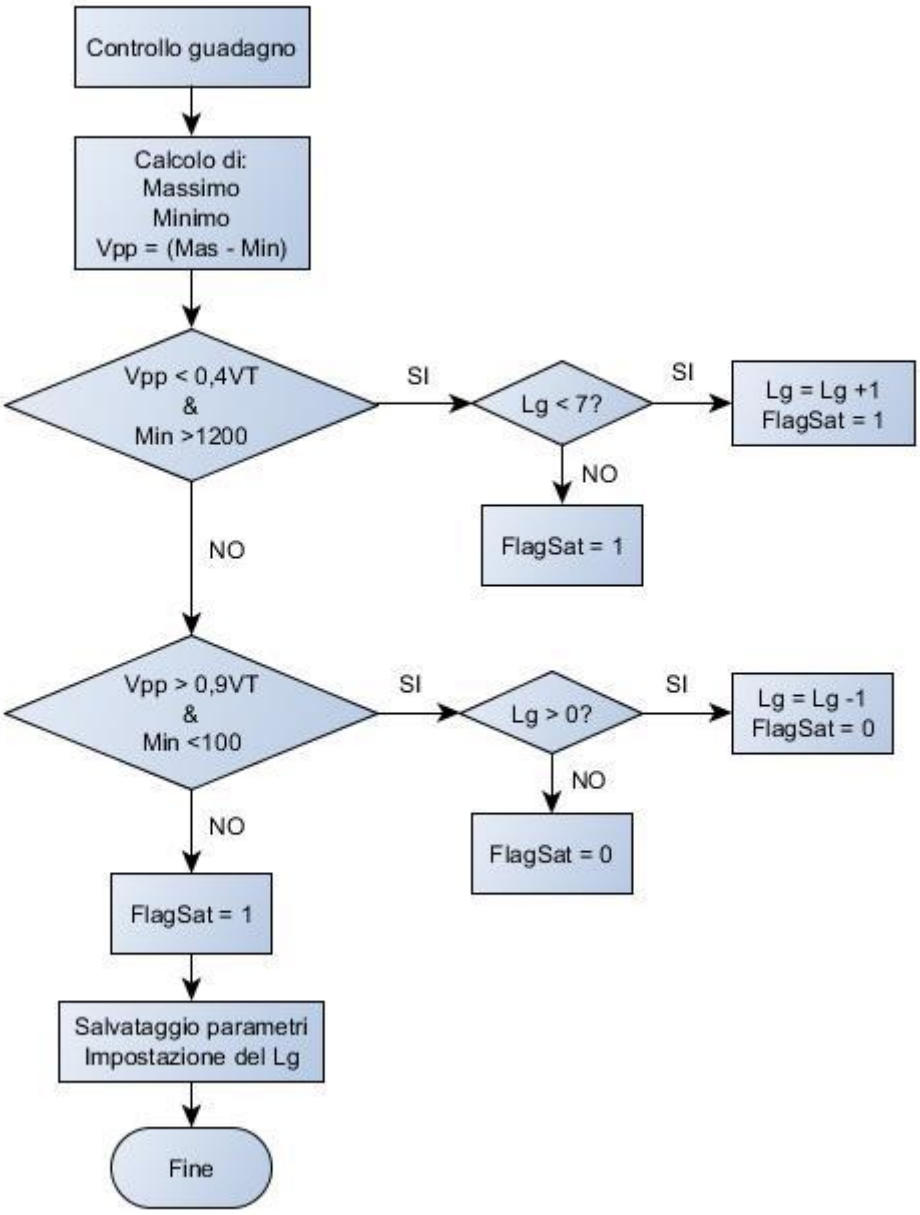


Figura 18: Controllo guadagno senza incmax

Questo controllo si fa per ogni frame di campioni. Per il corretto analisi posteriore o futuro dei campioni si deve sapere il guadagno impostato mentre si faceva il campionamento, il canale e se si deve scartare il frame perche ha saturato il segnale. Per questa ragione, ogni volta che si fa il controllo, si salvano questi tre dati dei quali i due ultimi sono flags.

Calcolo di Vrms

Per ottenere il valore di Vrms, si aplica la funzione:

$$V_{rms} = \lim_{T\to\infty} \sqrt{\frac{1}{T}\int_0^T [f(t)]^2 dt}$$

Come si ha una funzione discreta invece di continua, la formula utilizata è:

$$V_{rms} = \sqrt{\frac{\sum V^2 \cdot w^2}{300}}$$

dove V è il valore del campione e w il valore corrispondente della ventane di pesatura. Come l'ingresso di Arduino è solo positivo, realmente, il segnale preso non sta centrato in 0, ha una componente di continua che si deve togliere. Inoltre, il valore quadratico medio ottenuto così si riferisce al valore quadratico medio dei valori dei campioni (tra 0 e 4096), ma per avere il dato in unità di corrente, cioè, in Volts, si deve trasformare il dato.

Per ultimo, il valore ottenuto si deve moltiplicare per un fattore 1,581 perchè la finestra di pesatura atenua il segnale.

Alla fine la formula utilizata è:

$$V_{rms} = 1{,}581 \cdot \frac{3{,}3}{4096}\sqrt{\frac{\sum (V-\bar{V})^2 \cdot w^2}{300}}$$

Quindi, l'algoritmo è:

- Sumare tutti i valori del vettore, del frame.
- Fare la divisione della suma tra la dimensione del vettore per ottenere il valore medio, cioè aprossimativamente il valore della componente di continua del segnale.
- Fare il summatorio di: $\sum (V-\bar{V})^2 \cdot w^2$
- Fare la radice quadra del valore ottenuto nel paso anteriore.
- Finalmente moltiplicare l'ultimo valore per il fattore di correzione e per la fraccione per avere il risultato in volts.

Una volta ottenuto il valore Vrms, si deve separare in parte reale e parte decimale per salvarlo nella SD perchè la comunicazione con la scheda si deve fare in Bytes.

Quindi, il diagramma di flusso del modo di funzionamento di calibrazione è quello mostrato a continuazione:

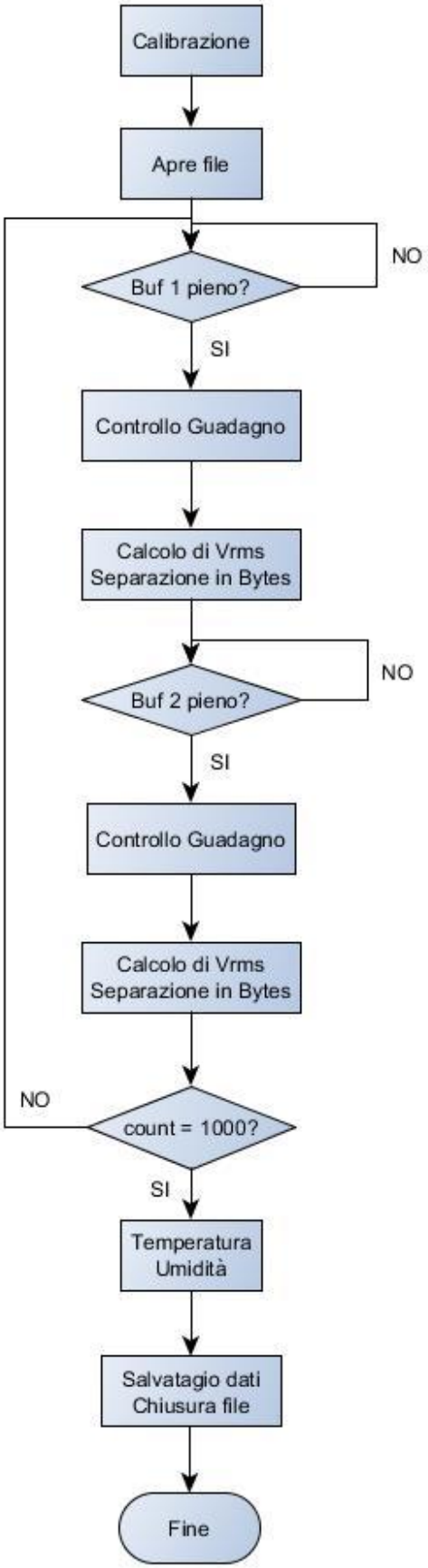


Figura 19: Calibrazione parametri

Come comprobazione, alla fine di questo processo, si stampano i dati salvati sul schermo. Nella calibrazione non c'è bisogno di calcolare la frequenza fondamentale, lo scopo di questo modo è trovare la relazione tra SPL e SAL, no caratterizzare il segnale o fare nessun altro analisi.

Monitoraggio

Nel modo di monitoraggio si fanno i calcoli spiegati nella calibrazione e anche quello del calcolo della F0. Il calcolo del valore di Vrms in questo modo non si fa per la stessa ragione che si fa nella calibrazione. Mentre nell'altro modo si faceva con l'obiettivo di trovare la relazione tra i valori di livello di accelerazione della pelle (SAL) presi per l'accelerometro e i livelli di pressione sonora (SPL), presi per il microfono d'aria, in questo modo, l'obiettivo è semplicemente quello di caratterizzare l'ambiente dove si fa il monitoraggio con lo scopo di poter analizzare nel futuro se c'è relazione tra lo sforzo fatto nel parlato per il soggetto sotto registrazione e le caratteristiche dell'ambiente in cui si trova.

Calcolo di F0

Come già riferito, ci sono diversi metodi per ottenere il valore della frequenza fondamentale di un segnale discreto, come l'algoritmo FFT o la correlazione, ma dovuto al carico computazionale, il metodo scelto è la correlazione. Questo metodo consiste nel mettere la onda su se stessa e moverla con l'obiettivo di trovare il punto di coincidenza. Ogni volta che si muove una posizione, si moltiplicano i valori sovrapposti e si fa il summatorio di loro. Tra questi valor si cerca il massimo relativo e il massimo assoluto, che è il prima chiamato punto di coincidenza.

Una volta trovato, e con l'informazione della frequenza di campionamento, si può calcolare senza problemi il periodo e per tanto, la frequenza naturale. Il problema è che non si tratta di un segnale completamente periodico, ma di un segnale con la stessa frequenza fondamentale (o molto simile) e diversi componenti armonici. Per questa ragione si trovano più di un massimo nella correlazione e si deve cercare il massimo assoluto di loro mettendo attenzione che sia la fondamentale, cioè, la più bassa, non un multiplo di questa; ma non quella corrispondente ad un periodo nullo dove ovviamente il segnale coincide al 100% con se stesso.

Per evitare questi non si inizia la correlazione nella prima posizione ma più avanti e non si prendono tutti i massimi relativi trovati, ma solo i primi, per trovare veramente la frequenza fondamentale. Il valore salvato è il valore del indice della posizione del vettore dove si è trovato il massimo della correlazione. Il valore della frequenza in unità di frequenza (Hz) si calcola dopo con matlab.

Quindi, l'algoritmo per il calcolo della F0 è quello mostrato nel diagramma di flusso:

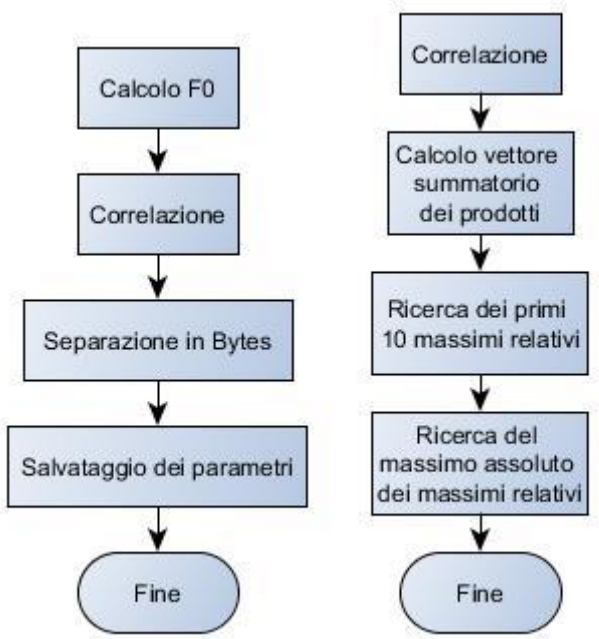


Figura 20: Calcolo di correlazione e F0

Come si lavora con un segnale discreto, il massimo della correlazione si può trovare soltanto in posizioni discrete, in valori di periodo discreti. Per cercare di trovare il vero massimo che magari non corrisponde a una posizione esatta, si salvano anche il valore previo e posteriore a questo massimo e cosí poter trovare l'equazione che gli une e calcolare la posizione del massimo della funzione, il periodo e così la frequenza. L'analisi sarebbe fatto con matlab.

In ogni caso, i valori salvati si devono separare in Bytes per la comunicazione con la scheda SD.

Il salvataggio dei valori (parametri e guadagno) si fa per ogni frame. Si salva un file ogni 3 minuti di forma continua, cioè, si apre un file, si inizia a salvare i valori dei parametri dei frame e dopo tre minuti si chiude e si apre uno nuovo. I valori di temperatura e umidità si prendono una volta al minuto, cioè, tre volte per file, abbastanza per la caratterizzazione dell'ambiente. Anche se i valori si prendono ogni minuto, si salvano insieme alla fine di ogni file. Aggiungendo questo all'algoritmo, si ha il seguente diagramma finale per il calcolo della frequenza fondamentale:

Figura 21: Calcolo F0

Finalmente, il diagramma di flusso del modo di monitoraggio è quello mostrato a continuazione, composto per gli algoritmi di calcolo dei parametri, di controllo di guadagno, l'ottenimiento dei valori di temperatura e umidità, il salvataggio dei valori, l'apertura e chiusura dei file...

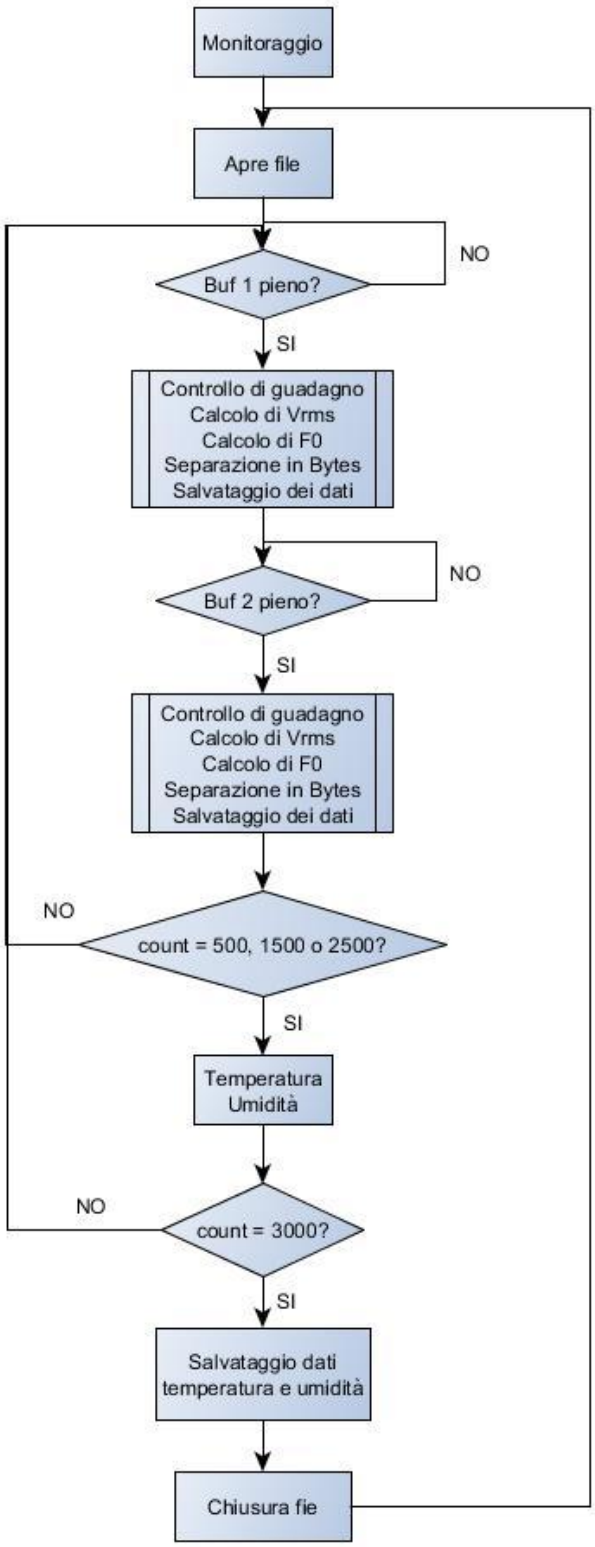


Figura 22: Monitoraggio parametri

3.3.4 Programma di salvataggio dei dati grezzi

Nel caso del programma dei dati grezzi, l'unica differenza tra il modo di calibrazione e quello di monitoraggio è il tempo di funzionamento, cioè, mentre nel primo modo, si salva soltanto un file di dati, nel secondo, si salvano dei file finchè non si spegne o ferma il dispositivo.

Calibrazione

Come si ha visto prima, la calibrazione è necessaria per trovare la relazione tra i SAL e i SPL. In questo programma semplicemente si salvano i dati grezzi senza fare nessun calcolo. Come si ha spiegato prima il salvataggio dei dati si deve fare con Bytes. L'ADC della scheda ha una risoluzione di 12 bits, quindi i campioni presi hanno questa dimensione. Per poter salvare tutta l'informazione quello che si fa è separare ogni campione preso in due Bytes, uno con i bits più significativi e l'altro con i meno significativi. Una volta separato ogni valore del frame in un altro vettore, si salva questo vettore nella SD. Questo processo si fa per ogni canale per i due buffer (con la tenica commentata dei due buffer per il campionamento continuo nel tempo) un totale di 20 volte facendo per ognuno di loro il controllo di guadagno spiegato per l'altro programma.

Quando si è arrivato alla fine, e si hanno salvato tutti i dati si prendono i valori di temperatura e umidità e si salvano alla fine. Di questa forma si ha una caratterizzazione più completa dell'ambiente. Come accade con i dati provenienti del convertitore analogico-digitale, anche i dati di umiditá e temperatura hanno una dimensione maggiore di un Byte perche sono dati decimali. In questo caso si salvano anche come due Bytes essendo uno la parte entera e l'altro la parte decimale. A continuazione si mostra il diagramma di flusso correspondente.

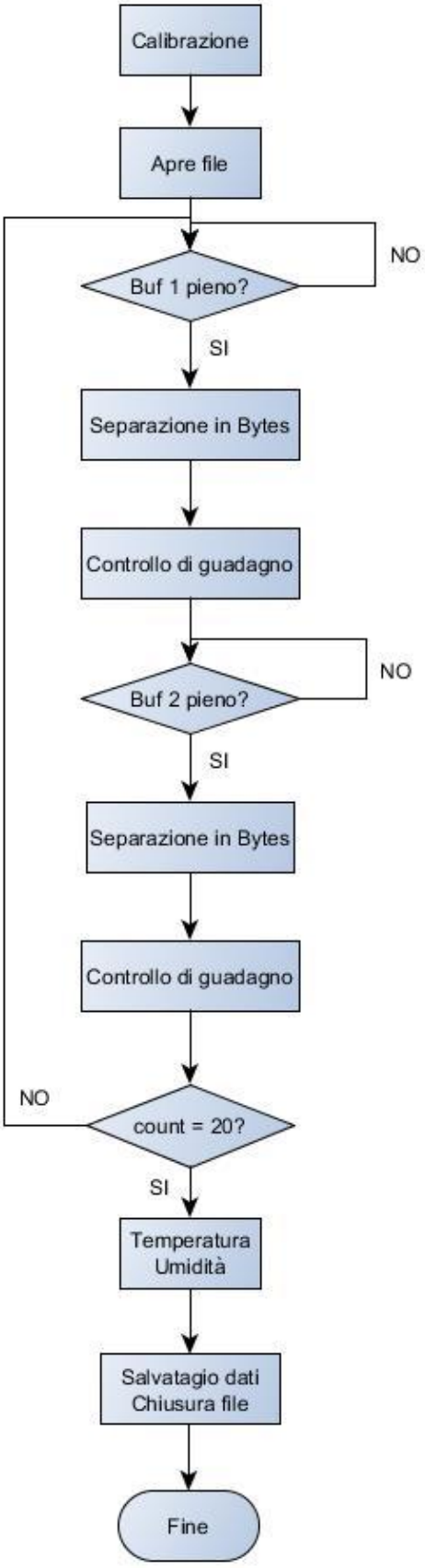


Figura 23: Calibrazione dati grezzi

Monitoraggio

Nel monitoraggio, si fa più meno lo stesso ma di forma continua finche non si ferma il processo per l'utente. In questo modo di funzionamento si deve lavorare con più di un file per l'enorme quantità di dati a salvare. Come nel monitoraggio del programma dei parametri, si apre e chiude un file ogni 3 minuti e mentre nella calibrazione si prendono i valori di temperatura e umidità una sola volta, in questo modo, questa informazione si prende tre volte per file, cioè, una volta al minuto. E dovuto al funzionamento continuo, non si può fare il salvataggio dei campioni una sola volta come nel modo di calibrazione, si fa per ogni frame. I valori di temperatura e umidità si salvano alla fine di ogni file. Come sempre, si fa il controllo del guadagno per ogni frame e si salvano i parametri corrispondenti.

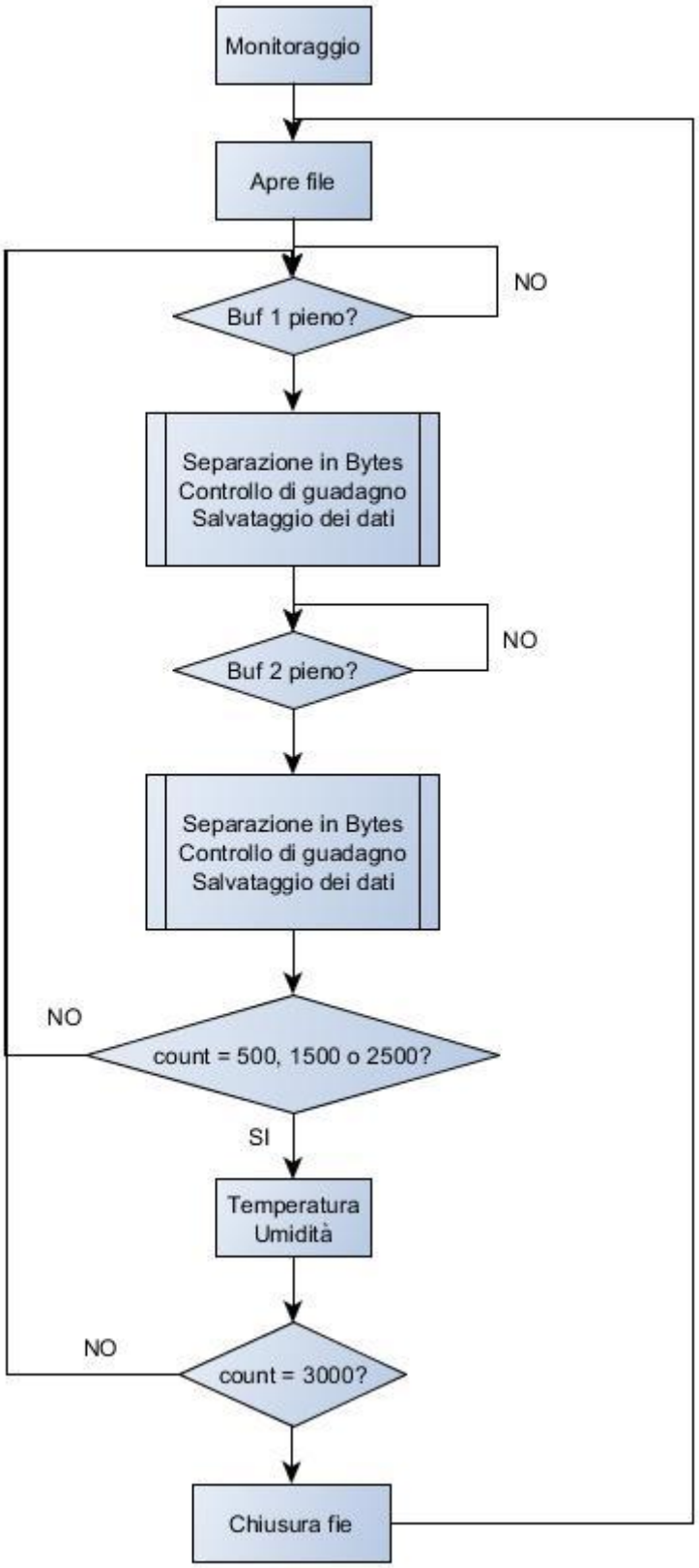


Figura 24: Monitoraggio dati grezzi

4. Sperimentazione e risultati

I testi realizati si possono dividere in quelli che servono a verificare il corretto funzionamento del dispositivo sviluppato e quelli finali che mostrano un esempio dell'informazione che si ottiene nel suo normale utilizzo:

4.1 Verifica funzionamento Sensore di temperatura e umidità.

In questo caso, il test consiste nel mettere il sensore di temperatura e umidità in una camara climatica nella quale si possono impostare le condizioni desiderate. Le condizioni normali di utilizzo sono quelli che si possono trovare all'interno di un edificio. Per questa ragione, le temperature del test vano dei 20 gradi ai 35, temperatura già abbastanza alta per una stanza climatizzata; e i valori di umidità del 30 al 80%. Nelle figure si può vedere la camera, la disposizione del sensore all'interno e il panel di controllo per impostare i valori desiderati.

Figura 25: Camera climatica

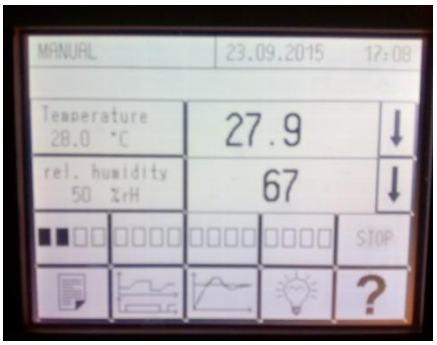


Figura 26: Panel di controllo della camera climatica

Il test si divide in due, per primo si fa un test a umidità costante (50%) e temperatura variabile:

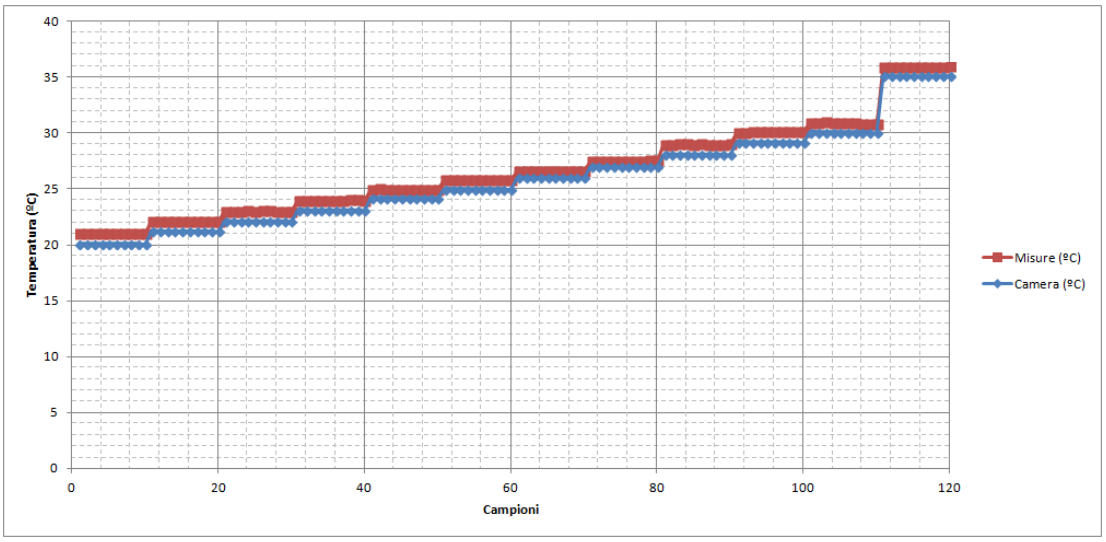


Figura 27: Test temperatura e umidità

Analisi delle differenze tra i valori della camera e quelli misurati per il sensore:

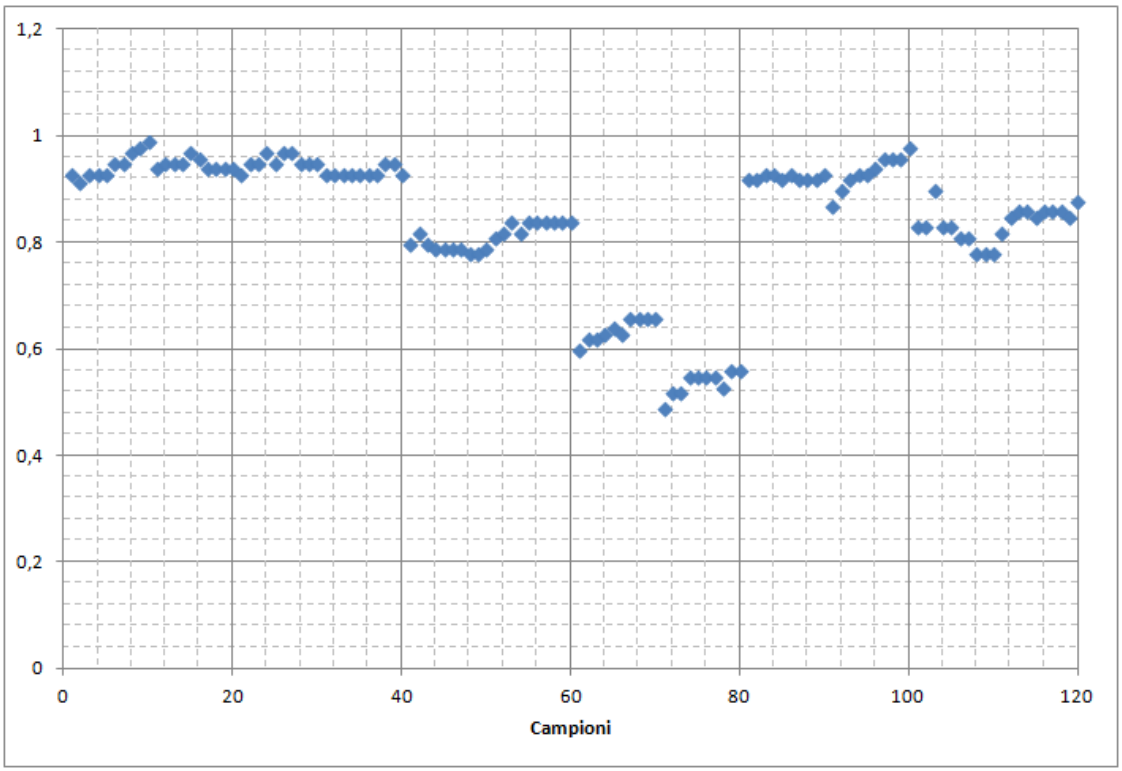


Figura 28: Errori test temperatura

Tabella 2: Errori test temperatura

Misura media	Valore di confronto	Errore massimo (ºC)	Percentuale del erroe massimo	Errore medio (ºC)	Percentuale del erroe medio	Errore standard
21,047	20,1	0,99	4,93%	0,95	4,71%	1,00
22,148	21,2	0,97	4,58%	0,95	4,47%	1,00
23,054	22,1	0,97	4,39%	0,95	4,32%	1,01
24,034	23,1	0,95	4,11%	0,93	4,04%	0,98
24,993	24,2	0,82	3,39%	0,79	3,28%	0,84
25,833	25	0,84	3,36%	0,83	3,33%	0,88
26,638	26	0,66	2,54%	0,64	2,45%	0,67
27,538	27	0,56	2,07%	0,54	1,99%	0,57
29,024	28,1	0,93	3,31%	0,92	3,29%	0,97
30,135	29,2	0,98	3,36%	0,94	3,20%	0,99
30,918	30,1	0,90	2,99%	0,82	2,72%	0,86
35,955	35,1	0,88	2,51%	0,86	2,44%	0,90

La seconda parte del test si fa a temperatura costante (25ºC) e umidità variabile:

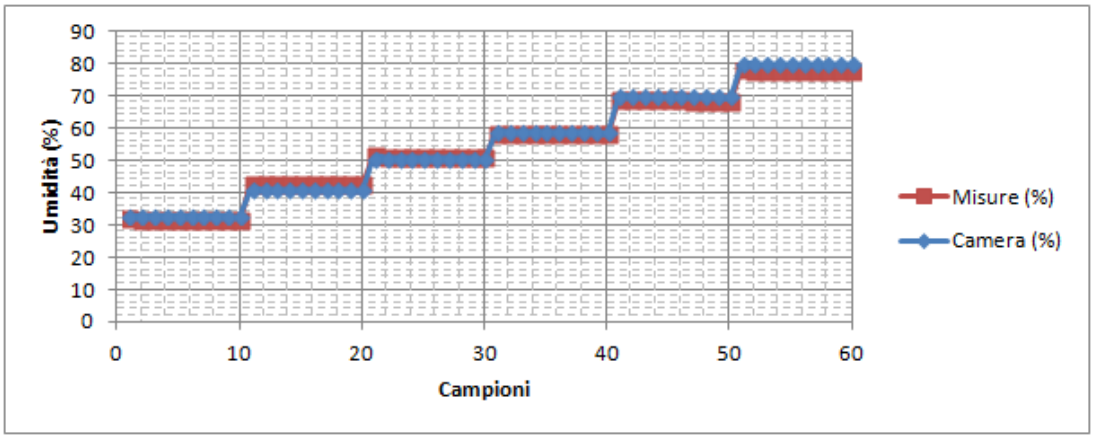


Figura 29: Test umidità

Analisi delle differenze tra i valori della camera e quelli misurati per il sensore:

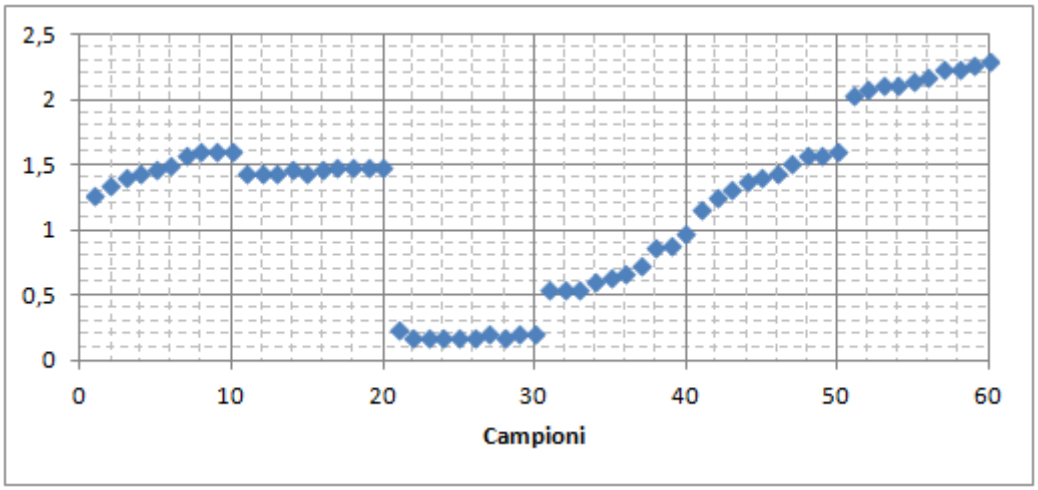


Figura 30: Errori test umidità

Tabella 3: Errori test umidità

Misura media	Valore di confronto	Errore massimo (%)	Percentuale di errore massimo	Errore medio (%)	Percentuale di errore medio	Errore standard
31,513	33	1,61	4,88%	1,49	4,51%	1,57
42,47	41	1,50	3,66%	1,47	3,59%	1,55
51,195	51	0,24	0,47%	0,20	0,38%	0,21
58,287	59	0,99	1,68%	0,71	1,21%	0,77
68,569	70	1,62	2,31%	1,43	2,04%	1,52
77,821	80	2,30	2,88%	2,18	2,72%	2,30

Come si può vedere nell'analisi, la differenza media nel caso della temperatura è di un grado e nel caso dell'umidità le differenze vano da meno da 0,5 a quasi 2,2; comunque, in entrambi casi rappresentano sempre meno del 5% del valore. Queste differenze possono essere dovute a due cause:

- Il sensore non ha la coperta che fa un poco di isolamento e così nel caso della temperatura prende valori maggiori e nel caso dell'umidità minori.
- Le dimensioni della camera no permettono un valore costante nello spazio e la posizione del sensore della propria camera ha delle condizioni un po diverse.

4.2 Calcolo della costante del microfono d'aria.

Per ottenere questo valore, si deve confrontare il valore di Vrms calcolato, col valore in dB preso per il sonometro come si vede nella figura 31.

Figura 31: Calcolo costante del microfono

Come si può vedere, l'esperimento consiste nel mettere una sorgente di rumore costante (in questo caso una talk box) di fronte al sonometro e al microfono d'aria.

La misura del sonometro è di 75,7 dB e il valore di tensione preso per il microfono è: 0,036V con un livello di guadagno 7,cioè, un guadagno di 128. Il valore del microfono senza amplificazione sarebbe: (0,036V/128) = 0,28mV.

Quindi, il valore della costante è:

$$75dB = 20log\left(\frac{P_{ref}}{P_0}\right)$$

$$3{,}75 = \log(P_{ref}) - \log(P_0)$$

$$P_{ref} = 109{,}05$$

$$P_{ref} = K_{mic} \cdot Vrms_{mic}$$

$$K_{mic} = \frac{P_{ref}}{Vrms_{mic}} = \frac{109{,}05}{0{,}00028125} = 4{,}017 \cdot 10^{12}$$

L'obiettivo di questo test è ottenere la costante che servirà per trovare la relazione tra i valore di tensione del microfono a contatto che rappresenta il SAL e il valore de tensione del microfono d'aria che rappresenta il SPL.

4.3 Calcolo di Vrms

Per realizzare questo test, si ha utilizzato un generatore di segnale digitale collegato come ingresso, con il segnale della vocale 'a' registrato previamente nel generatore. All'uscita del generatore si collega anche un multimetro per misurare il valore di Vrms direttamente dalla fonte e avere così un confronto al valore del parametro calcolato per il programma. Il test si fa variando ampiezza e frequenza del segnale. Come la misura del multimetro si fa direttamente all'uscita del generatore, non ha ancora l'amplificazione, quindi il valore calcolato dovrà essere uguale al valore del multimetro moltiplicato per il guadagno. La ragione di usare questo segnale generato è quella di conoscerlo e così avere un confronto. Il test si realizza con diversi valori di frequenza e ampiezza ma nelle prossime figure si mostrano solo i test fatti a 100Hz e a 250Hz come mostra rappresentativa visto che i risultati sono molto simili come si aspettava.
Frequenza fissa a 100Hz e ampiezza variabile:

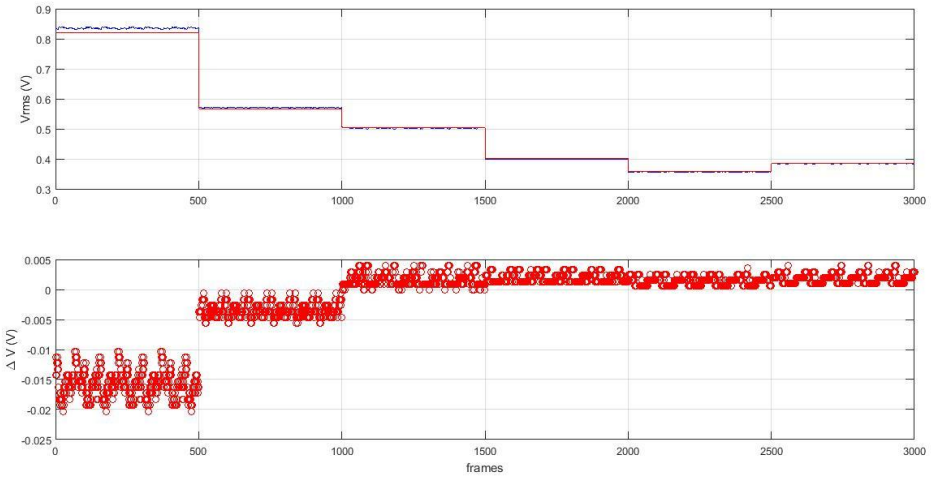


Figura 32: Test 3 a 100Hz

Nella prima grafica si mostra il confronto del valore calcolato (blu) e il valore del multimetro (rosso). Nella seconda, si inizia l'analisi delle differenze che continua nella tabella 4.

Tabella 4: Errori test 3 a 100Hz

Misura media	Valore di confronto	Errore massimo (mV)	Percentuale di errore massimo	Errore medio (mV)	Percentuale di errore medio	Errore standard
0,83548	0,819712	20,29	2,48%	15,77	1,92%	15,93
0,570834	0,567328	5,67	1,00%	3,51	0,62%	3,69
0,502242	0,503944	3,94	0,78%	1,71	0,34%	1,98
0,399314	0,40132	3,32	0,83%	2,01	0,50%	2,15
0,356124	0,35756	3,56	1,00%	1,44	0,40%	1,62
0,384144	0,385952	3,95	1,02%	1,81	0,47%	1,97

Frequenza fissa a 250Hz e ampiezza variabile:

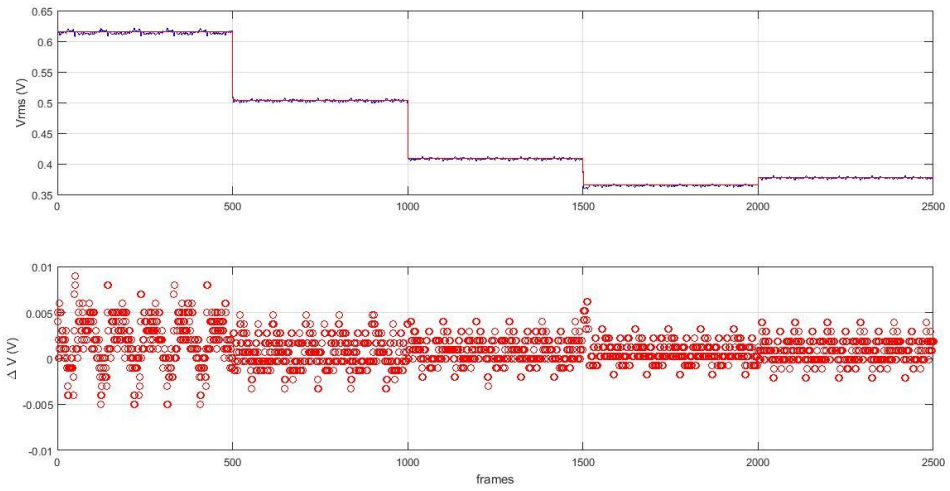


Figura 33: Errori test 3 a 250Hz

Come nel caso di 100Hz, nella prima grafica si mostra il confronto del valore calcolato (blu) e il valore del multimetro (rosso), e, nella seconda, si inizia l'analisi delle differenze che continua nella seguente tabella.

Tabella 5: Errori test 3 a 250Hz

Misura media	Valore di confronto	Errore massimo (mV)	Percentuale di errore massimo	Errore medio (mV)	Percentuale di errore medio (mV)	Errore standard
0,61411	0,616	9,00	1,46%	2,59	0,42%	3,20
0,503058	0,503704	4,70	0,93%	1,38	0,27%	1,74
0,408086	0,408972	3,97	0,97%	1,22	0,30%	1,61
0,364428	0,365206	6,21	1,70%	1,13	0,31%	1,54
0,377068	0,377891	3,89	1,03%	1,14	0,30%	1,48

In entrambi casi, gli errori sono molto bassi e possono essere dovuti al valore di confronto del multimetro perchè mentre l'altro valore si registra automaticamente, l'altro deve essere preso a mano e non avere registro delle piccole variazioni.

4.4 Calcolo di F0

Come nel caso del test anteriore, si ha utilizzato lo stesso segnale generato per lo stesso generatore. Di questa forma si ha il confronto del valore calcolato direttamente con il valore di frequenza impostato nel generatore. Il test si ha fatto variando la frequenza e lasciando l'ampiezza fissa.

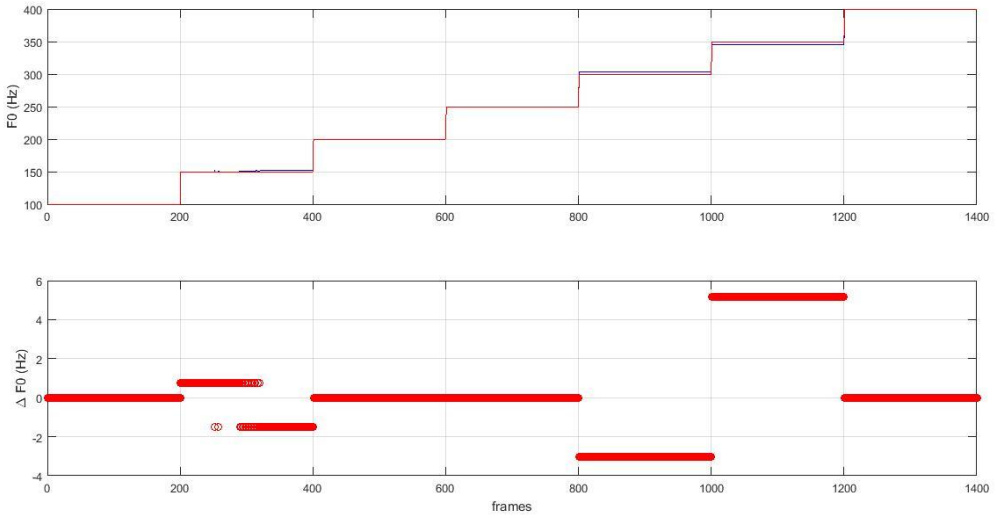


Figura 34: Errori calcolo F0

Tabella 6: Errori calcolo F0

Misura media	Valore di confronto	Errore massimo (Hz)	Percentuale di errore massimo	Errore medio (Hz)	Percentuale di errore medio	Errore standard
100	100	0,00	0,00%	0,00	0,00%	0,00
150,373134	150	1,52	1,01%	1,13	0,75%	1,19
200	200	0,00	0,00%	0,00	0,00%	0,00
250	250	0,00	0,00%	0,00	0,00%	0,00
303,030303	300	3,03	1,01%	3,03	1,01%	3,04
344,827586	350	5,17	1,48%	5,17	1,48%	5,19
400	400	0,00	0,00%	0,00	0,00%	0,00

Come si vede nell'analisi, di nuovo gli errori sono molto bassi, trascurabili e sono dovuti al carattere discreto del segnale. Come si ha spiegato nella sezione dell'implementazione, il calcolo della frequenza fondamentale si fa con l'autocorrelazione, si trova la posizione dove ha il suo valore massimo ma come si tratta di un segnale discreto, non si può trovare il massimo tra due posizioni.

Si deve commentare anche il problema dell'algoritmo. Si deve trovare la quantità di massimi relativi tra i quali si cerca il assoluto per avere il valore giusto del periodo e non un multiplo che fa ottenere una frequenza naturale più bassa della reale. Nel caso di questo progetto, si ha scelto il valore per avere un calcolo corretto della F0 nel rango di frequenze (100Hz - 400Hz) che corrisponde alla voce umana.

4.5 Sensibilità di F0

Lo scopo di questo test è trovare la sensibilità del dispositivo ai cambiamenti di frequenza. Il test consiste nella variazione della frequenza di forma continuata e costante (cioè, una rampa) per tre minuti.

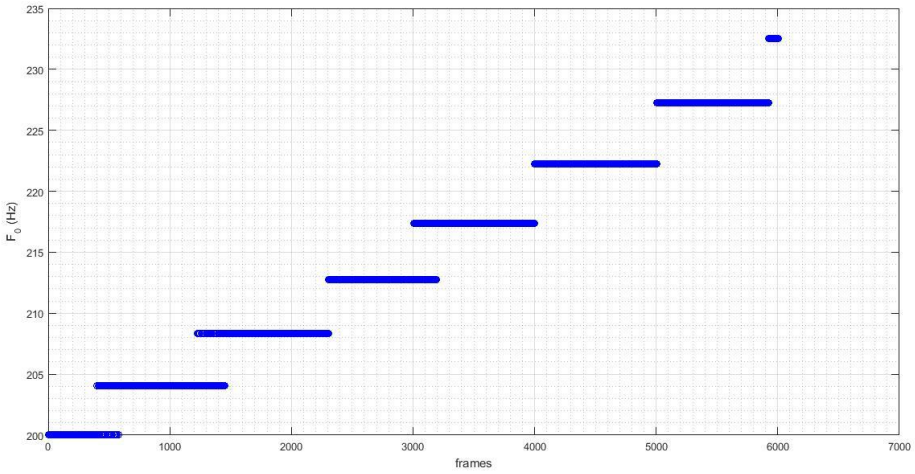


Figura 35: Sensibilità F0

Osservando i risultati dei valori ottenuti di F0, si può dire che la sensibilità del sistema si trova tra 4 e 5Hz. La sensibilità si deve, come gli errori del test anteriore, al carattere discreto del segnale. Se si guardano i valori dei periodi corrispondenti ad ogni valore di F0 calcolato per il sistema, si vede che sono posizioni consecutive.

4.6 Controllo di saturazione

Dopo avere verificato il corretto funzionamento dei diversi componenti del dispositivo, si fa un test per vedere il segnale reale della voce e verificare il corretto funzionamento anche dell'algoritmo di controllo del guadagno. Fin questo momento di comprobazione con un segnale reale, il controllo non aveva nessun problema. Il test si fa con parlato e con una probba simile alla di taratura, cioè, con la vocale 'a' a diverse intensità cercando di arrivare a impostare tutti i guadagni.

Queste prime immagine corrispondono quindi, all'algoritmo prima di fare nessun cambiamento.

4.6.1 Test con la vocale 'a' a diverse intensità:

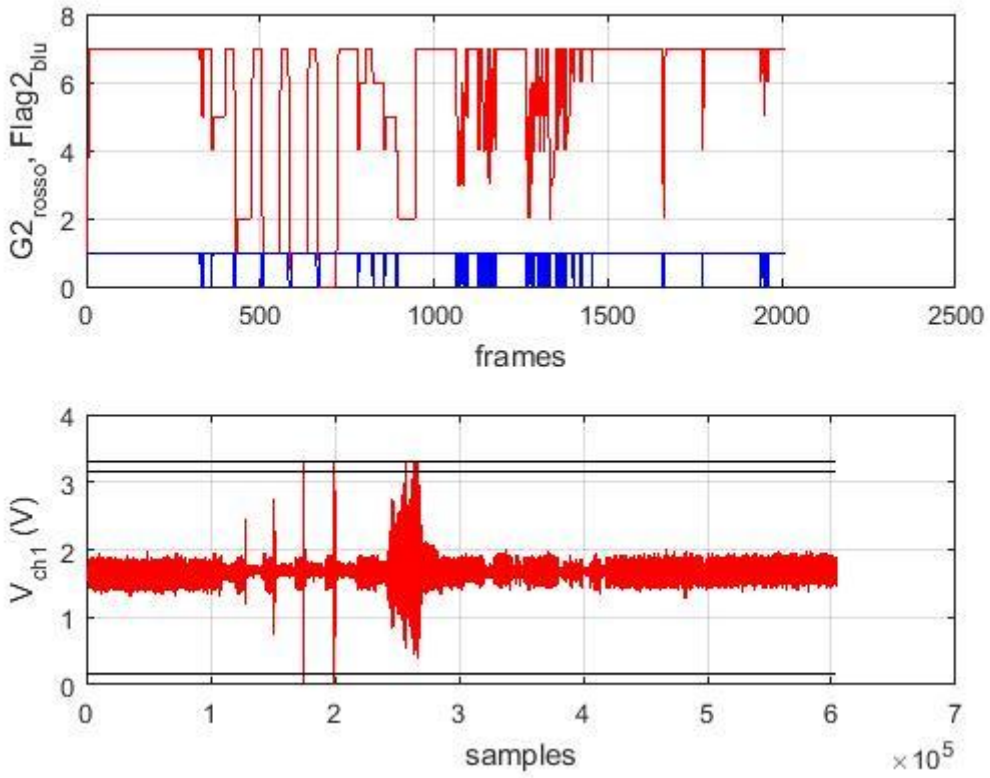
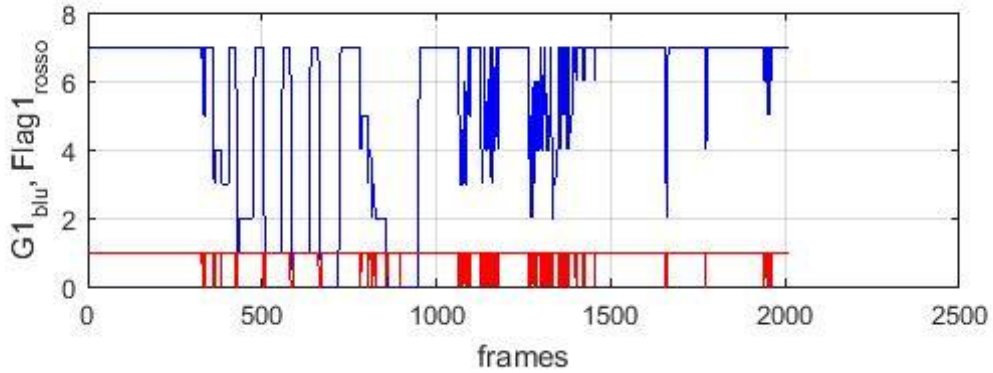


Figura 36: Test 6.1 microfono d'aria

La figura 36 è del canale del microfono d'aria. Nel primo grafico si mostra il guadagno e il flag di saturazione, essendo saturato quando è uguale a '0'. Nel secondo grafico, si mostrano i valori di tensione del segnale, i dati grezzi di campionamento. Le righe nere mostrano i limiti di saturazione e sono impostate a 0.15, 3.15 y 3.3. Il fondo di scala è 3,3V.



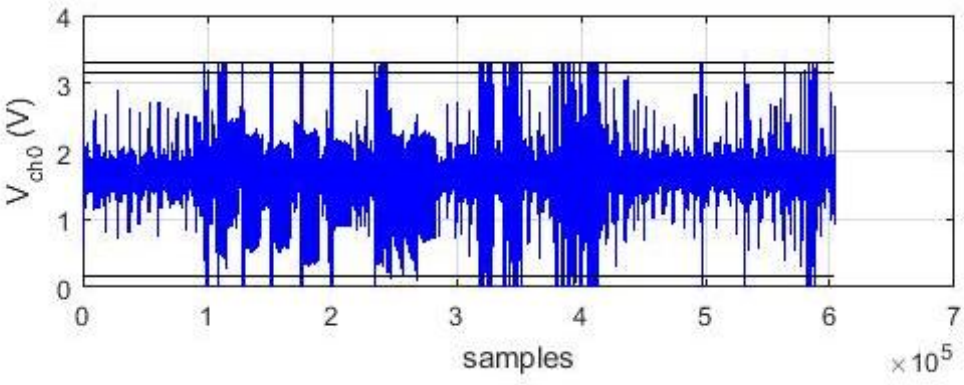


Figura 37: Test 6.1 microfono contatto

La figura 37 invece, è il canale del microfono a contatto. Come nella figura precedente, nel primo grafico si mostra il guadagno e il flag; e nel secondo i valori di tensione del segnale. Come si può vedere, il secondo canale satura molto di più. La saturazione del segnale del canale del microfono a contatto in queste condizioni, in questo test, satura un 6,80%.

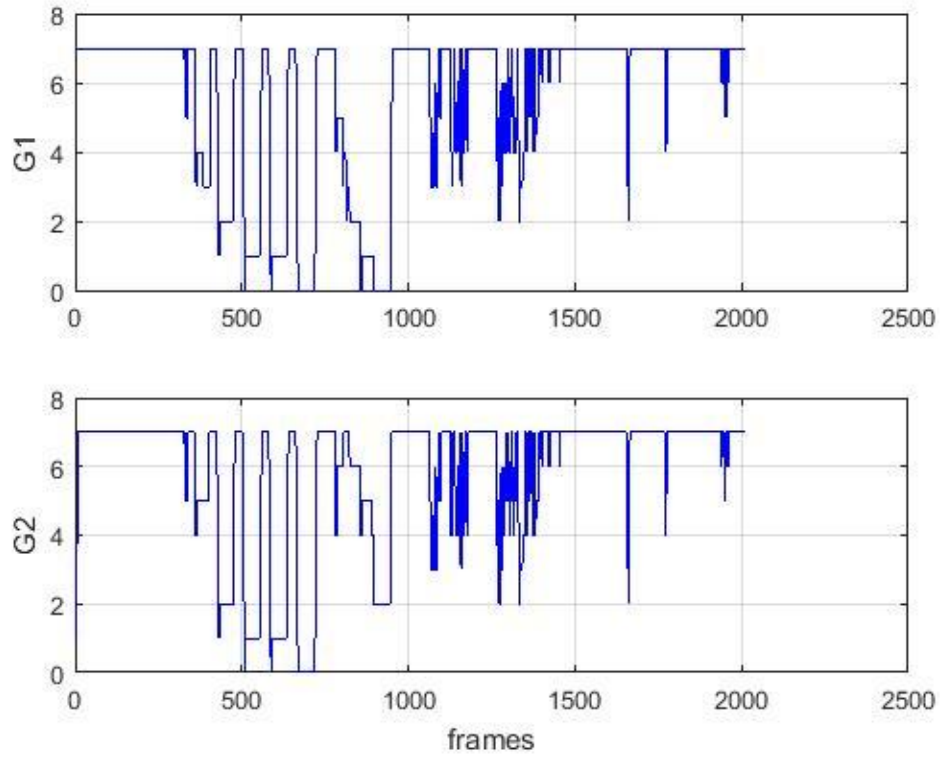


Figura 38: Test 6.1 guadagni

In questa ultima figura si mostrano i livelli di guadagno di entrambi canali. Si può osservare che il secondo canale ha livelli di guadagno più elevati.

4.6.2 Test di parlato

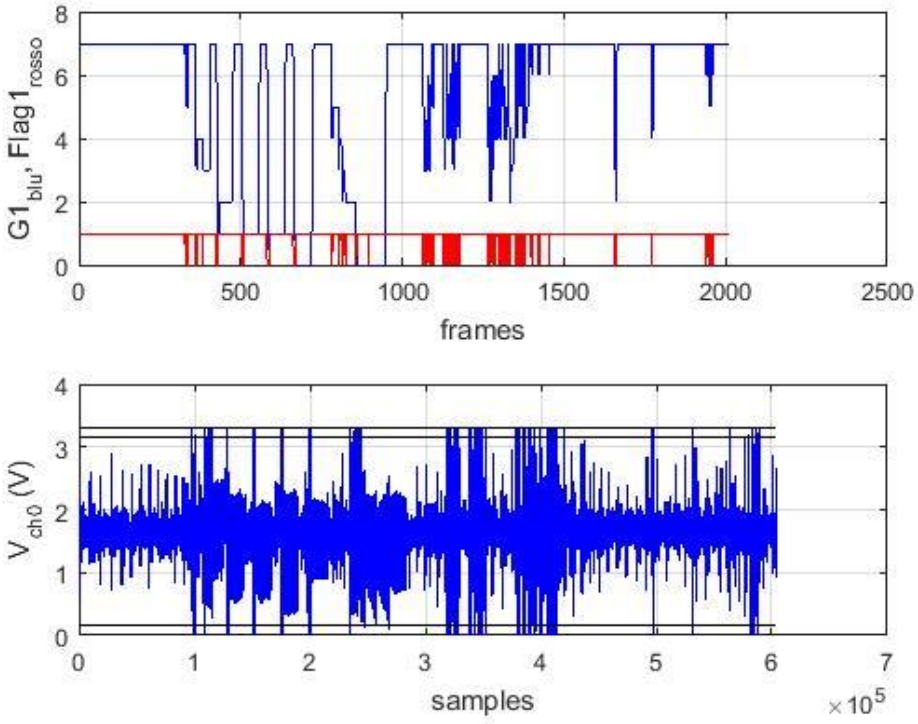


Figura 39: Test 6.2 microfono contatto

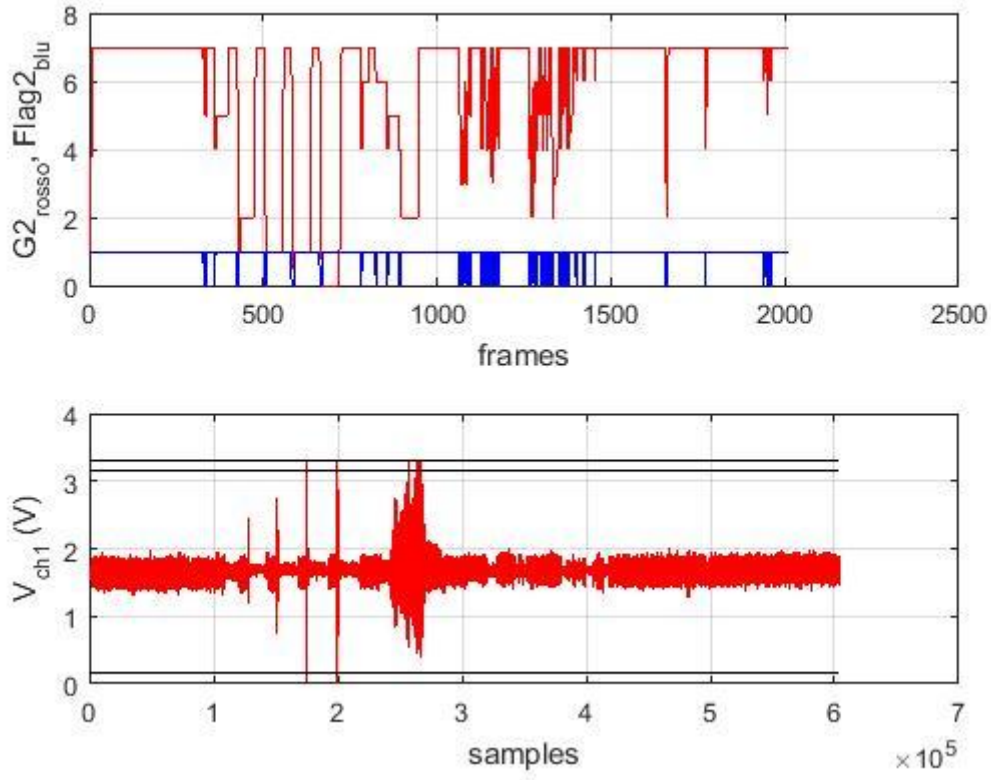


Figura 40: Test 6.2 microfono d'aria

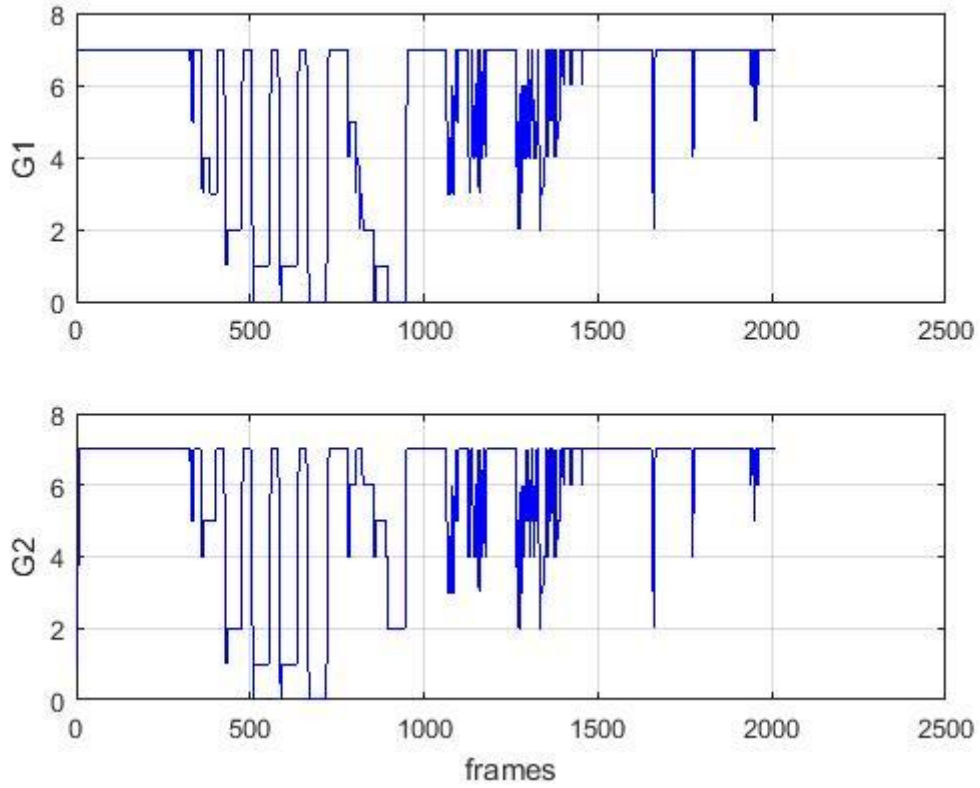


Figura 41: Test 6.2 guadagni

Le figure di quest test di parlato sono analoghe al test con la vocale. Cioè, la prima corrisponde al canale del microfono d'aria (livello di guadagno, flag di saturazione e segnale grezzo), la seconda al microfono a contatto (stessi variabili) e la terza ai livelli di guadagno di entrambi canali.

In questo caso si osserva che la saturazione e molto maggiore che nel test con la vocale arrivando a un 24,1638%, una quarta parte. Questo si deve alle pause del parlato. Ogni volta che si produce un piccolo silenzio si arriva a guadagno 7 di forma che quando si inizia di nuovo a parlare il segnale satura. Per abbassare la percentuale di saturazione si decide di rallentare la salita del livello di guadagno. Si introduce una nuova variabile chiamata 'incmax' che rappresenta quante volte consecutive si deve voler alzare il livello di guadagno prima che si faccia. Cioé, finche non si hanno 'incmax' frames consecutivi con una ampiezza massima di tensione minore del 40% del fondo di scala, nons si alza il livello di guadagno. Come si ha visto il problema c'è sul canale del microfono a contatto soprattutto ed è su questo canale che si rallenta la salita. Sul canale del microfono d'aria non si aggiunge questa modifica per due motivi: Non ha tanti problemi di saturazione ed è un canale che si vuole per caratterizzare l'ambiente in cui si trova il soggetto sotto test, non è un problema se si perde qualche frame perche anche così si ha informazione abbastanza per la caratterizzazione.

A continuazione si mostrano i risultati della ripetizione del test per entrambi casi (vocale e parlato) con diversi valori della variabile 'incmax' per osservare come cambia la percentuale di saturazione (PS) con i diversi valori della variabile. Le prime prove si possono pensare come prove con incmax=1 per l'analisi dei risultati.

Vocale:

incmax=3 PS= 2,2545%: incmax=4 PS= 4,6477%:

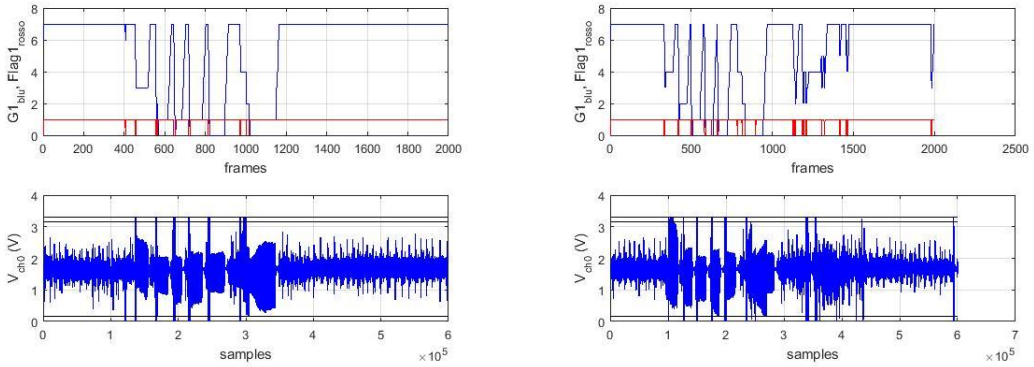


Figura 42: Vocale incmax 3 e 4

incmax=5 PS= 4,3697%: incmax=6 PS= 2,2795%:

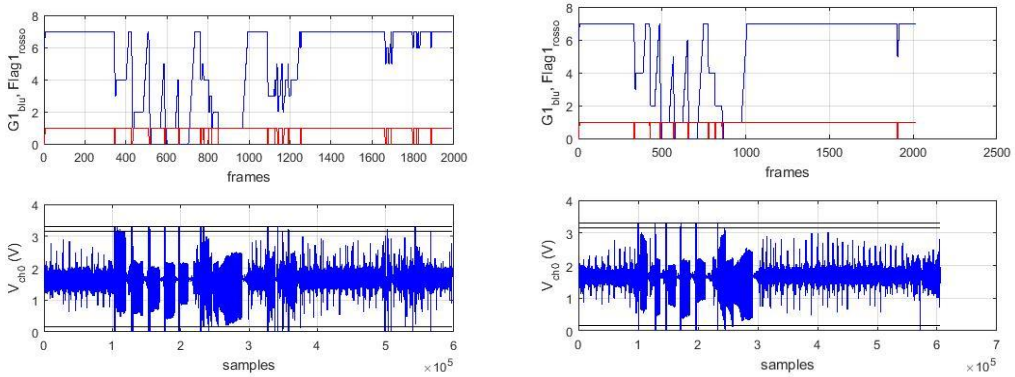


Figura 43: Vocale incmax 5 e 6

Parlato:

incmax=3 PS= 14,4483%: incmax=4 PS= 4,4084%:

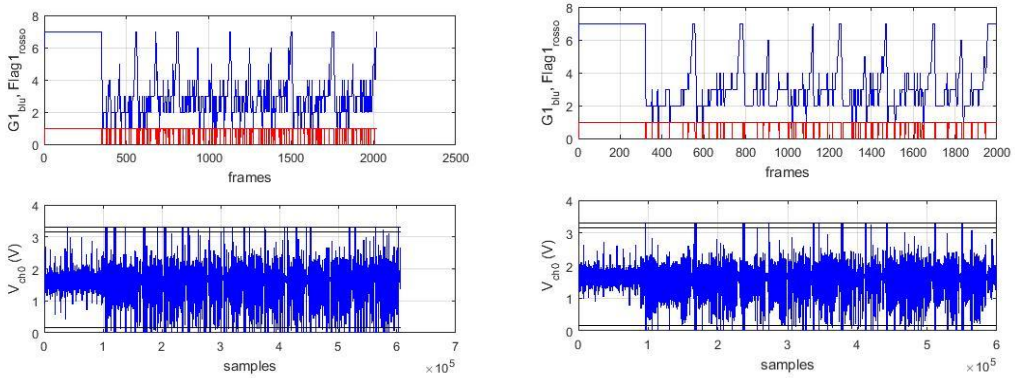


Figura 44: Parlato incmax 3 e 4

incmax=5 PS= 7,4612%: incmax=6 PS= 6,6165%:

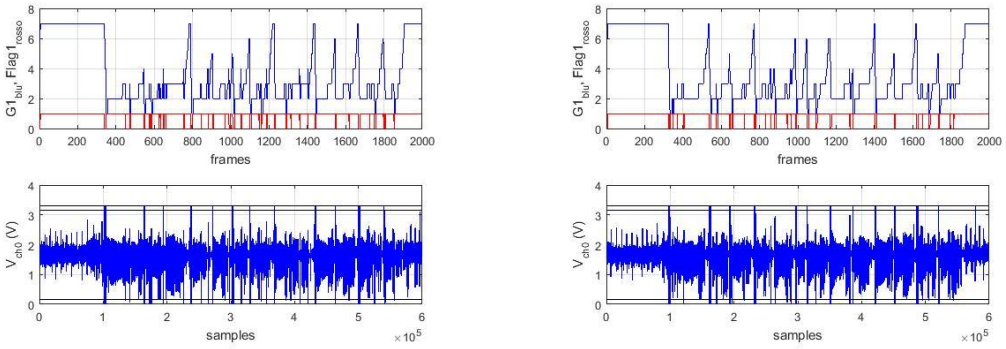


Figura 45: Parlato incmax 5 e 6

Analisi dei risultati:

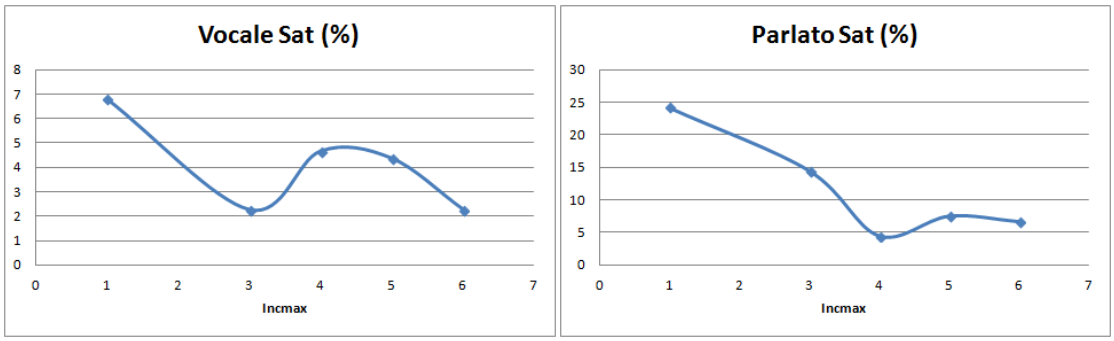


Figura 46: Saturazione Vocale e Parlato

Queste grafiche mostrano il cambiamento della saturazione in funzione del valore di 'incmax'. Di questa informazione si può trarre che questo cambiamento ha importanza soprattutto nel test di parlato come è naturale ma che comunque, in entrambi casi si abbassa considerablemente la percentuale di saturazione. La differenza tra di loro è dovuta alle caratteristiche dei test. Nel parlato si fanno molte più pause piccole che avevano come risultato la saturazione, mentre nel test della vocale le pause erano soltanto tre o quatro. In entrambi casi si può vedere una tendenza chiara anche se c'è un punto che sembra di non seguirla.

Quindi, guardando il test di parlato che si avvicina di più a una situazione reale, si può dire che il abbassamento della saturazione e maggiore nei primi valori della variabile e minore nei ultimi e che a partire di un valore 4 si può dire sotto il 10%, un valore accettabile, si perdono frames ma non troppi.

Si deve comunque tenere conto delle condizioni dei test in questa analisi, non si ha lo stesso livello di saturazione con voci diverse e i test non erano uguali al 100%, il parlato era aleatorio e i test della vocale anche se si vogliono fare tutti uguali, è molto dificile. Per le caratteristiche del soggetto sotto test in questo caso, si potrebbe dire che riesce a raggiungere livelli di saturazione più alti di altri soggetti.

Per eliminare la variabile del parlato aleatorio si fa di nuovo il test con la lettura del testo di una canzone con i seguenti risultati:

incmax=4 PS= 8,45%: incmax=5 PS= 8%:

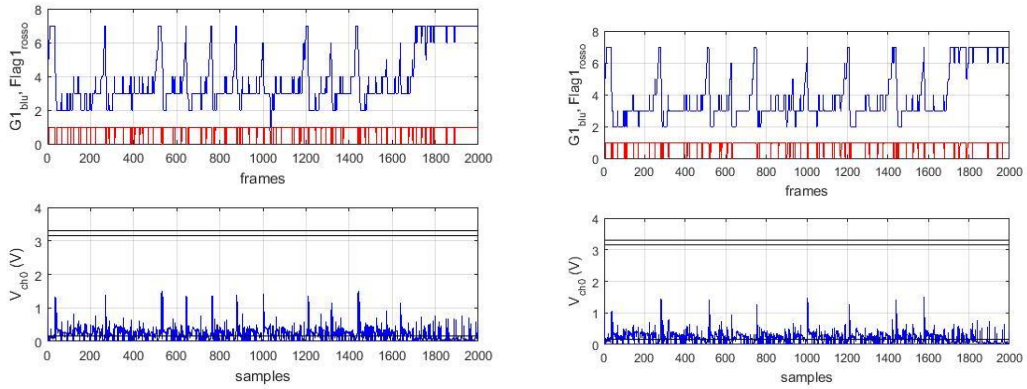


Figura 47: Parametri Parlato incmax 4 e 5

incmax=6 PS= 9,6048%: Analisi:

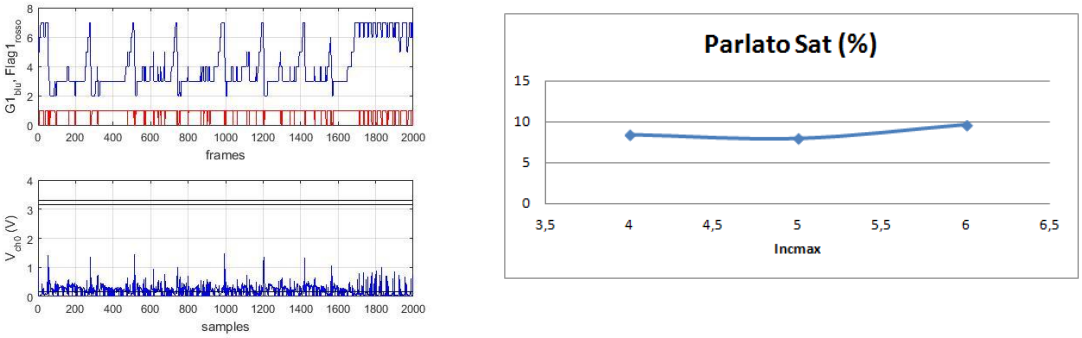


Figura 48: Parametri Parlato incmax 6 e Analisi

4.7 Taratura e verifica

L'obiettivo di questo test insieme al secondo test, è trovare la relazione tra i valori di SAL dell'accelerometro e i valori di SPL in unità acustiche (dB). Per raggiungere l'obiettivo si utilizza il valore della costante del microfono e il test di taratura con un valore de incmax=5. Detto test si fa impostando il microfono d'aria ad una distanza costante e conosciuta (12 cm) come si vede nella figura 49.

Figura 49: Test 7

Con questa configurazione si fa il test di taratura che consiste nel emettere un suono a diverse intensità (in questo caso la vocale 'a') ottiendo il seguente risultato di Vrms di entrambi canali:

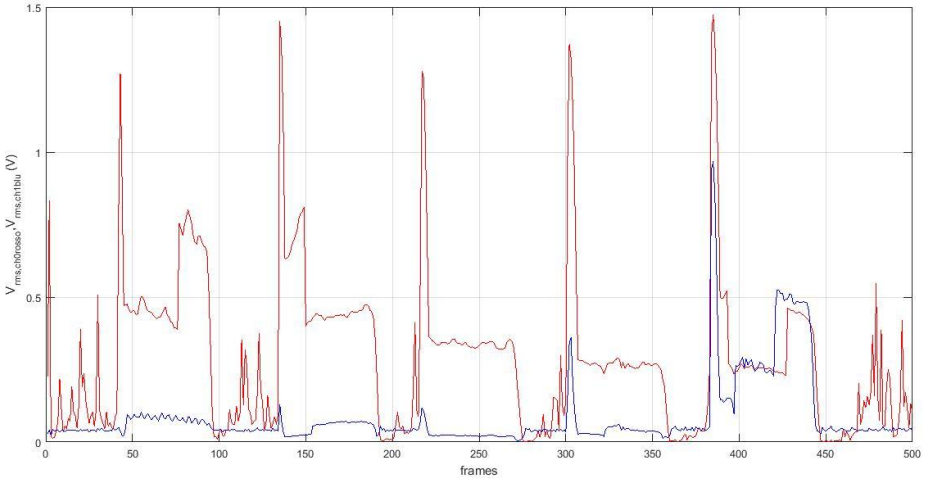


Figura 50: Taratura

Dove il rosso è il primo canale, quello del microfono a contatto e il blu il microfono d'aria. Prima di fare l'analisi si deve togliere a questi valori di tensione il guadagno, per questa ragione è importante salvare anche questa informazione.

Una volta ottenuti i valori di Vrms prima della sua amplificazione, introducendo il valore della costante del microfono trovata, si può calcolare il SPL del microfono d'aria con la seguente formula:

$$SPL_{ref} = 20log\left(\frac{P_{ref}}{P_0}\right)$$

dove Po e una costante acustica e:

$$P_{ref} = K_{mic} \cdot Vrms_{mic}$$

E cercare di trovare una relazione tra questi valori e i valori di Vrms del primo canale come si vede nella seguente figura:

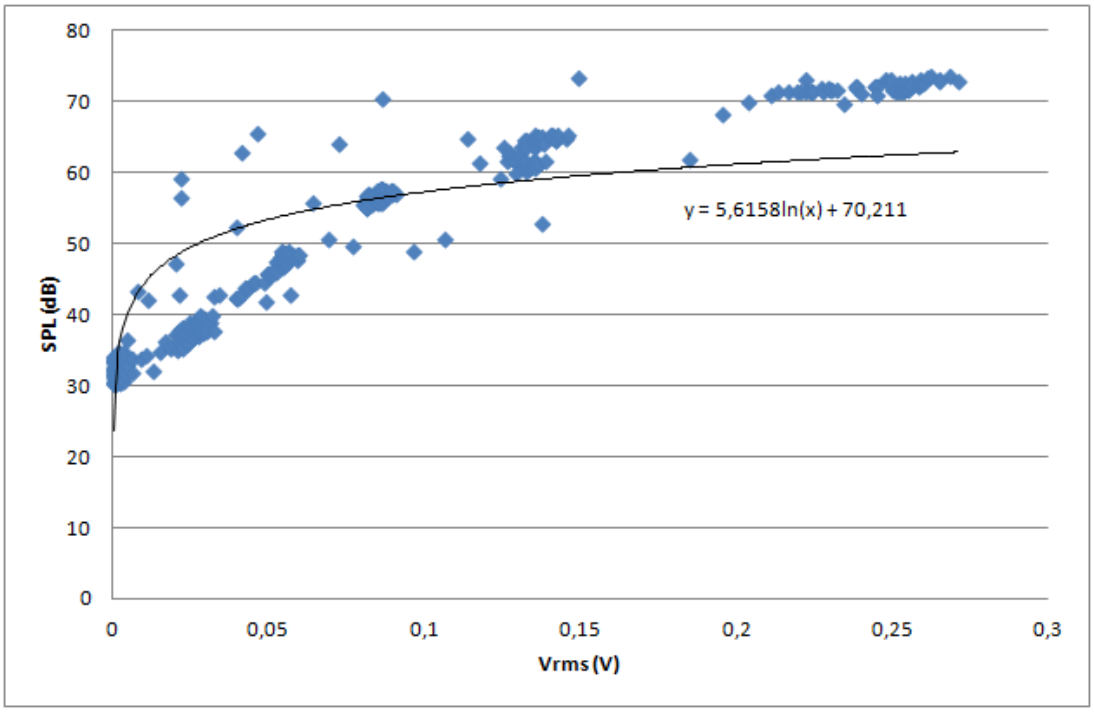


Figura 51: Verifica

Per fare la verifica di questa relazione, si calcola ora i valori di SPL corrispondenti ai valori di Vrms del microfono a contatto secondo la relazione trovata:

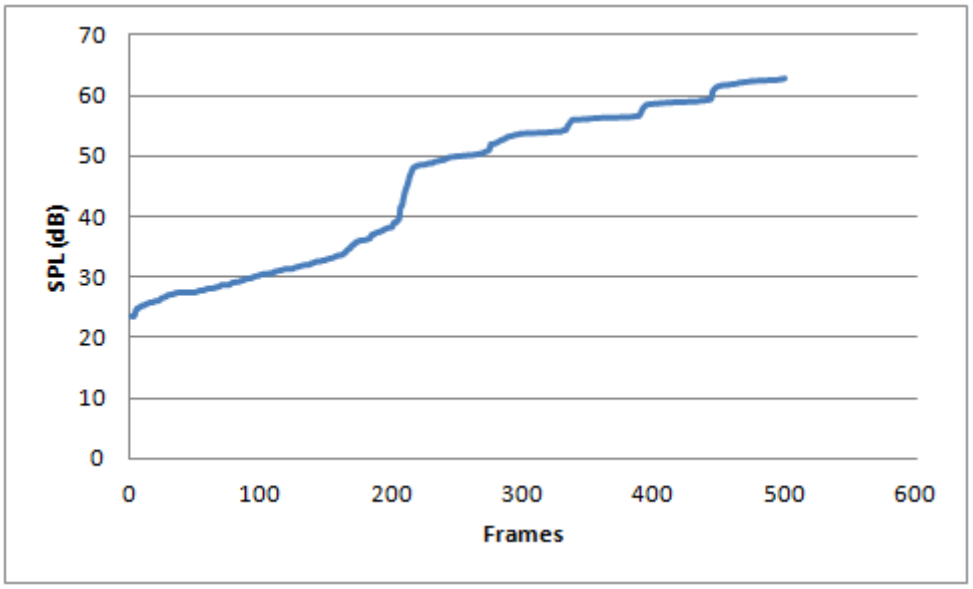


Figura 52: Verfica

E si fa l'analisi delle differenze tra i valori di SPL:

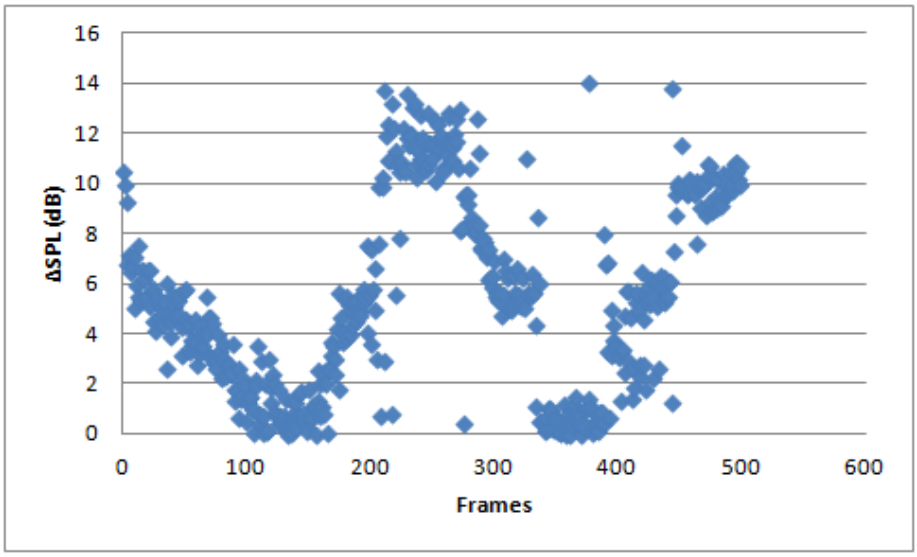


Figura 53: Errori taratura e verifica

L'errore medio è: 5,4 dB, troppo alto perche una differenza di 3dB significa un suono con il doppio livello di pressione. Questi risultati possono essere dovuti a un ambiente poco adeguato per fare il test, sarebbe meglio averlo fatto nella camera anecoica, a errori umani e si potrebbe avere ripetuto il test per ottenere la costante del microfono più di una volta e così avere un valore più sicuro.

4.8 Monitoraggio

Finalmente, dopo avere verificato tutto il funzionamento del dispositivo si fa una prova di monitoraggio che consiste nel leggere il testo della canzone e calcolare i parametri spiegati nel progetto.

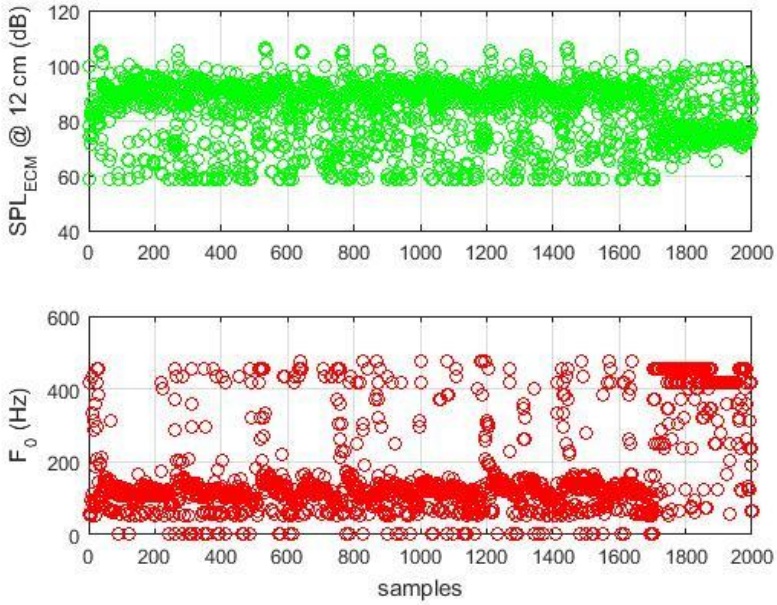


Figura 54: SPL e F0 monitoraggio

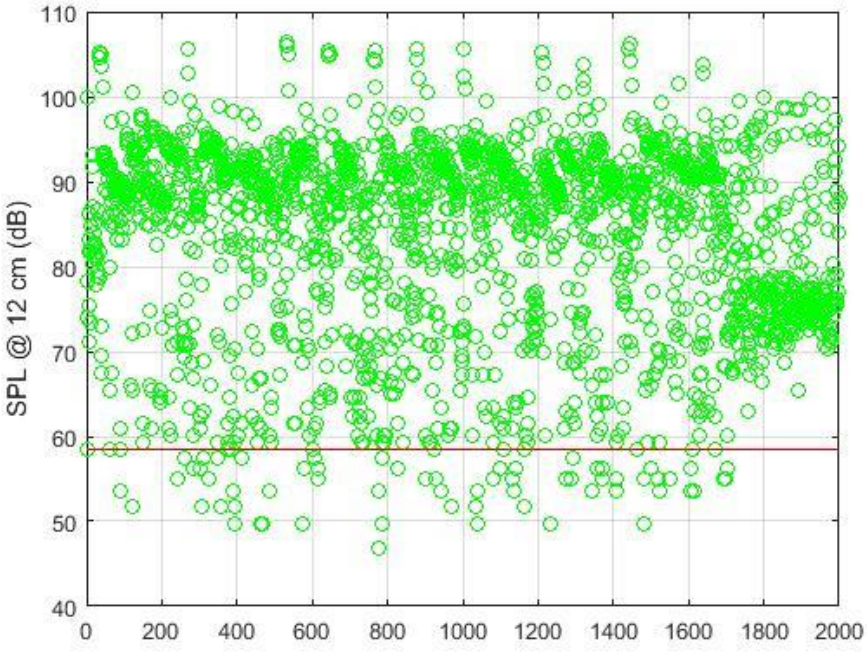


Figura 55: Dosado monitoraggio

In questa ultima figura si mostra il dosado di parlato, cioè, sotto la riga rossa, sono campioni che corrispondono a frames di silenzio, di pausa. Il suo valore è dovuto a movimenti della gola, rumore elettrico del microfono...

La percentuale di parlato quindi è maggiore dal 80%. Un valore troppo alto ma si deve tener conto che il test consiste nella lettura di un testo, non è una situazione reale di parlato. È per tanto un risultato consistente.

5. Conclusioni e miglioramenti futuri

5.1 Conclusioni

Come si aspettava i risultati mostrano un corretto sviluppo del dispositivo. Tutti i miglioramenti sono stati implementati.

Si hanno lasciato soltanto due canali in entrambi modi di funzionamento, si ha aggiunto il sensore di temperatura e umidità con una comunicazione digitale con arduino e per ultimo si hanno messo anche dei amplificatori a guadagno programmabile.

I risultati dei test in generale sono consistenti e per tanto accettabili. In quelli che sono un poco meno buoni si hanno spiegato le cause e riguardano più al come si hanno fatto le prove che a un funzionamento scorretto e si ha spiegato anche come si possono migliorare.

5.2 Miglioramenti futuri

I miglioramenti futuri ruotano intorno ad una idea principale: Il problema del carico computazionale. Dopo aver aggiunto più funzionalità, si inizia ad intravedere problemi nel funzionamento in tempo reale per causa del carico computazionale. Per questa ragione e per risparmiare risorse si pensano i seguenti possibili miglioramenti:

- Se si tratta di un frame saturato, cioè per buttare via, non fare i calcoli dei parametri. Questo solo per il modo di monitoraggio, nel modo di calibrazione fare i calcoli e salvare tutto per avere una registrazione completa del funzionamento del dispositivo.
- L'arduino due ha la possibilità di comunicare con il protocollo SPI soltando con tre dispositivi, se c'è bisogno di questa comunicazione nel futuro non ci sono più SS. Alcuni dei dispositivi collegati così si possono collegare anche ai pin digitali direttamente lasciando SS liberi.
- Come si ha visto, parte del carico computazionale si può fare per hardware risparmiando tempo per migliorare gli algoritmi oppure aggiungere funzionalità.

6. Bibliografia

[1] Ingo R. Titze, Principles of voice production, UniversityIowa and the Denver Center of the performing arts.
[2] L. R. Rabiner and R. W. Schafer, Digital Processing of Speech Signals
[3] Ingo R. Titze, Peter S. Popolo, Estimation of sound pressure levels of voiced speech from skin vibration of the neck.
[4] A. Carullo, A. Vallan, A. Astolfi, A low-cost platform for Voice Monitoring.
[5] Ingo R. Titze, Peter S. Popolo, Jan G. Svec, Measurement of vocal doses in speech: experimental procedure and signal processing.
[6] Ingo R. Titze, Peter S. Popolo, Jan G. Svec, Enric Hunter, Karen Rogge-Miller, The calibration and Setup of the NCVS Dosimeter.
[7] Ingo R. Titze, Peter S. Popolo, Jan G. Svec, Karen Rogge-Miller, Technical consierations in the Design of a Wereable Voice Dosimeter.
[8] Daryush D. Mehta, Matías Zañartu, Shengran W. Feng, Harold A. Cheyne, Robert E. Hillman, Mobile voice health monitoring using a wereable accelerometer sensor and a smartphone platform.
[9] Harold A. Cheyne, Helen M. Hanson, Ronald P. Genereux, Kenneth N. Stevens, Robert E. Hillman, Development and testing of a portable vocal accumulator.
[10] Fredric Lindström, Jonas Söholm, Magnus Berggren, A noise dosimeter system where the contribution from the werer's own voice is excluded.
[11] Peter S. Popolo, Jan G. Svec, Ingo R. Titze, Adaptation of a Pocket PC for use as a wereable voice dosimeter.
[12] Marcus Wirebrand, Real-time monitoring of voice characteristics using accelerometer and microphone measurements.
[13] Xu Gaoxiang, A low-cost micro-controller based device for online estimation of vocal parameters.
[14] ANSI S1.1-1994. American National Standard: Acoustic Terminology. Sec 3.03.
[15] Structural Acoustics & Vibration Technical Committee. "Structural Acoustics & Vibration Technical Committee". Retrieved 22 May 2013.

7. Allegato 1

Codice del programma dei parametri:

```
//-----------------------------------------------------------------------
#include <Array.h>    //Per ottenere il massimo, il minimo e Vpp
#include <SHT1xTH.h>
// Specify data and clock connections and instantiate SHT1x object
#define dataPin  11  // pins a cambiar porque ya estan utilizados para otras cosas en el
voice-care
#define clockPin 2  // pins a cambiar porque ya estan utilizados para otras cosas en el
voice-care
SHT1xTH sht1xTH(dataPin, clockPin);
//-----------------------------------------------------------------------
#include <SD.h>
#include <SPI.h>

const int size=300;
volatile uint16_t length=size;                 // buffer size
uint16_t ib = 0;                    // buffer index
uint32_t result;                   //value read from ADC

int buf1_0[size];
int buf1_1[size];
int buf2_0[size];
int buf2_1[size];
long buf_autoco_1[size];
long buf_autoco_2[size];
long buf_autoco_p[size];
int sft,auto_idx;
long ADCval;
long tstart;
long tstop;
char bufferName[10];
int count = 0;
int nameIndex = 1;                // index for saved file name
String fileName ="";
File myFile;
boolean calMode=0;
boolean cal_rms=1;
int chan=0;
int i,j,automax1,automax2;
long maxauto[10],maxauto1[10],maxautoidx[10],maxautoidx1[10];

int maxidx1;
byte MSD0[11];
byte MSD1[11];
byte autoval00;
byte autoval01;
byte autoval10;
byte autoval11;
byte autoval20;
byte autoval21;
float w[300];
float Vrms0[2000];
float Vrms1[2000];
float sum0=0;
float sum1=0;
float pi=3.141592653;
float vdc0=0;
float vdc1=0;
```

```
byte SD0[4000];
byte SD1[4000];
int SDidx=0;
int rms_idx=0;
float avr_Vrms0=0;
float avr_Vrms1=0;
float var_Vrms0=0;
float var_Vrms1=0;
byte SD0int;
byte SD1int;
byte SD0frac;
byte SD1frac;
//-------------------------------------------------------------
float temp_c;
float humidity;

byte TC[4];
byte HC[4];
byte T[8];
byte H[8];

byte tint;
byte tfloat;
byte hint;
byte hfloat;
int t;
//-------------------------------------------------------------
int Maxbuf0=0;
int Minbuf0=0;
int Maxbuf1=0;
int Minbuf1=0;
int Vpp0=0;
int Vpp1=0;
byte Gan0[3];
byte Gan1[3];
int VT=4096;
int inc=0;
int incmax=6;
//-------------------------------------------------------------

void setup() {
    Serial.begin(115200);
    pinMode(2,OUTPUT);    // port B pin 25
    analogWrite(2,255);   // sets up some other registers I haven't worked out yet
    pinMode(7,INPUT);
    pinMode(8,OUTPUT);
    pinMode(9,OUTPUT);
    digitalWrite(8,HIGH);
    //-------------------------------------------------------------
    SPI.begin(10);        //CS Amplificatore
    SPI.begin(52);
    //-------------------------------------------------------------
  Serial.println("Initializing SD card...");
    if (!SD.begin(4)){
      while(1){
        digitalWrite(9,LOW);
        digitalWrite(9,HIGH);
```

```
        digitalWrite(8,LOW);
        digitalWrite(9,HIGH);
    }
      }
for(i=0;i<300;i++)
{
 w[i] = 0.54 - 0.46*cos(2*pi*i/(length-1));
}
//-----------------------------
    Gan0[0]=6;
    Gan1[0]=6;
    Gan0[2]=0;
    Gan1[2]=1;
    SetPGA(Gan1[0],0,10);
    SetPGA(Gan0[0],0,52);

    TC[0]=0xFF;
    TC[3]=0xFF;
    HC[0]=0xFF;
    HC[3]=0xFF;
    T[0]=0xFF;
    T[7]=0xFF;
    H[0]=0xFF;
    H[7]=0xFF;
//-----------------------------
SD.remove("testData.log");
  if (SD.exists("testData.log")){
    Serial.println("testData.log exists.");
  }
  else {
    Serial.println("testData.log doesn't exist.");
  }

SD.remove("Monitor.log");
  if (SD.exists("Monitor.log")){
    Serial.println("Monitor.log exists.");
  }
  else {
    Serial.println("Monitor.log doesn't exist.");
  }

    if (!digitalRead(7)){
        calMode=0;
        digitalWrite(9,HIGH);
        Serial.println("Monitoring mode");
        myFile = SD.open("L0.log", FILE_WRITE);

    }else{
        calMode=1;
        digitalWrite(8,HIGH);
        Serial.println("CAL mode");
        myFile=SD.open("testData.log",FILE_WRITE);

    }
   setRegisters();
   Serial.println("set is ok");
}
```

```
void setRegisters()
{
 //WARNING
  //THE ARDUINO CHANNEL IS NOT THE SAME OF SAM PROCESSOR
  //ARDUINO SAM
  //AD7     CH0
  //AD0     CH7
  //AD6     CH1
  //AD1     CH6
  //AD2     CH5

      // Trigger Enable
      // Set TRGSEL TO 1: Timer Counter Channel 0 TIOA as trigger source
      // 12 bit resolution
      // normal mode
      // not use sequence registers
  REG_TC0_WPMR=0x54494D00; // enable write to registers of TC
  REG_TC0_CMR0=0b00000000000010011100010000000000; // set channel mode register (see
  datasheet)
  REG_TC0_RC0=2100; // counter period
  REG_TC0_RA0=1000;  // PWM value
  REG_TC0_CCR0=0b101;    // start counter
  REG_TC0_IER0=0b00010000; // enable interrupt on counter=rc
  REG_TC0_IDR0=0b11101111; // disable other interrupts
  NVIC_EnableIRQ(TC0_IRQn); // enable TC0 interrupts

  ADC->ADC_MR = 0x10310103;

  ADC->ADC_CHDR = 0xFFFF ;      // disable all channels
  adc_enable_channel(ADC, ADC_CHANNEL_7); //ADC A0 ON ARDUINO CORRESPOND CH7 ON SAM

  adc_enable_interrupt(ADC, ADC_IER_DRDY);

  //TIMER TRIGGER
  //adc_configure_trigger(ADC, ADC_TRIG_TIO_CH_0, 0);

  NVIC_EnableIRQ(ADC_IRQn);// enable ADC interrupts

  adc_start(ADC);
}


void TC0_Handler()
{
    long dummy=REG_TC0_SR0; // vital - reading this clears some flag
                            // otherwise you get infinite interrupts
}

void ADC_Handler(){
 if ((adc_get_status(ADC) & ADC_ISR_DRDY) == ADC_ISR_DRDY)
  {
        if (ib < length) {
          if(chan==0){
    result =adc_get_channel_value(ADC,ADC_CHANNEL_7);
```

```
            buf1_0[ib]= result;
            chan=1;
            ADC->ADC_CHDR = 0xFFFF ;      // disable all channels
            adc_enable_channel(ADC, ADC_CHANNEL_6);
                     }
         else if(chan==1){
    result =adc_get_channel_value(ADC,ADC_CHANNEL_6);
            buf1_1[ib++]= result;
            chan=0;
            ADC->ADC_CHDR = 0xFFFF ;      // disable all channels
            adc_enable_channel(ADC, ADC_CHANNEL_7);
         }
         }
         else if (ib >= length && ib < 2*length) {
           if(chan==0){
    result =adc_get_channel_value(ADC,ADC_CHANNEL_7);
            buf2_0[ib - length]= result;
            chan=1;
            ADC->ADC_CHDR = 0xFFFF ;      // disable all channels
            adc_enable_channel(ADC, ADC_CHANNEL_6);
          }
          else if(chan==1){
    result =adc_get_channel_value(ADC,ADC_CHANNEL_6);
            buf2_1[ib++ - length]= result;
            chan=0;
            ADC->ADC_CHDR = 0xFFFF ;      // disable all channels
            adc_enable_channel(ADC, ADC_CHANNEL_7);
          }
        } else if (ib==2*length) {
          if(chan==0){
    result =adc_get_channel_value(ADC,ADC_CHANNEL_7);
            buf1_0[0] = result;
            chan=1;
            ADC->ADC_CHDR = 0xFFFF ;      // disable all channels
            adc_enable_channel(ADC, ADC_CHANNEL_6);
              }
         else if(chan==1){
    result =adc_get_channel_value(ADC,ADC_CHANNEL_6);
            buf1_1[0] = result;
            chan=0;
            ADC->ADC_CHDR = 0xFFFF ;      // disable all channels
            adc_enable_channel(ADC, ADC_CHANNEL_7);
            ib=1;
           }
        }
}
    long ADCval = REG_ADC_LCDR;//vital to return to void loop!
}

void loop(){
if (calMode){
while (count<1000){

if(ib==length){  //buffer 1
              inc=CG(buf1_0, 0, inc);
              CG(buf1_1, 1, inc);
              for(j=0;j<size;j++)
```

```
            {
            vdc0=vdc0+buf1_0[j];
            vdc1=vdc1+buf1_1[j];
            }
            vdc0=vdc0/size;
            vdc1=vdc1/size;
            for(j=0;j<size;j++)
            {
            sum0 = sum0+(buf1_0[j]-vdc0)*w[j]*(buf1_0[j]-vdc0)*w[j];
            sum1 = sum1+(buf1_1[j]-vdc1)*w[j]*(buf1_1[j]-vdc1)*w[j];
            }
              Vrms0[rms_idx]=1.581*3.3*sqrt(sum0/length)/4096;
              Vrms1[rms_idx]=1.581*3.3*sqrt(sum1/length)/4096;
              sum0 = 0;
              sum1 = 0;
             vdc0=0;
             vdc1=0;
        SD0int=Vrms0[rms_idx]*10;
        SD0frac=Vrms0[rms_idx]*1000-SD0int*100;
        SD1int=Vrms1[rms_idx]*10;
        SD1frac=Vrms1[rms_idx]*1000-SD1int*100;
        rms_idx++;
        SD0[SDidx]=SD0int;
        SD1[SDidx++]=SD1int;
        SD0[SDidx]=SD0frac;
        SD1[SDidx++]=SD1frac;

//tstop=micros();
           }

else if(ib==2*length){//buffer 2
          inc=CG(buf2_0, 0, inc);
           CG(buf2_1, 1, inc);
            for(j=0;j<size;j++)
            {
            vdc0=vdc0+buf2_0[j];
            vdc1=vdc1+buf2_1[j];
            }
            vdc0=vdc0/size;
            vdc1=vdc1/size;
            for(j=0;j<size;j++)
            {
            sum0 = sum0+(buf2_0[j]-vdc0)*w[j]*(buf2_0[j]-vdc0)*w[j];
            sum1 = sum1+(buf2_1[j]-vdc1)*w[j]*(buf2_1[j]-vdc1)*w[j];
            }
              Vrms0[rms_idx]=1.581*3.3*sqrt(sum0/length)/4096;
              Vrms1[rms_idx]=1.581*3.3*sqrt(sum1/length)/4096;
              sum0 = 0;
              sum1 = 0;
             vdc0=0;
             vdc1=0;
        SD0int=Vrms0[rms_idx]*10;
        SD0frac=Vrms0[rms_idx]*1000-SD0int*100;
        SD1int=Vrms1[rms_idx]*10;
        SD1frac=Vrms1[rms_idx]*1000-SD1int*100;
        rms_idx++;
        SD0[SDidx]=SD0int;
```

```
         SD1[SDidx++]=SD1int;
         SD0[SDidx]=SD0frac;
         SD1[SDidx++]=SD1frac;
          Serial.println(count);
             count++;
            }
  }
adc_stop(ADC);
//--------------------------------------------------------------------------------
if (count==1000){
  Serial.println(count);
  temp_c = sht1xTH.readTemperatureC();
  humidity = sht1xTH.readHumidity();
  tint=temp_c;
  tfloat=temp_c*100-tint*100;
  TC[1]=tint;
  TC[2]=tfloat;
  hint=humidity;
  hfloat=humidity*100-hint*100;
  HC[1]=hint;
  HC[2]=hfloat;
  Serial.print(" Temperature: ");
  Serial.print(temp_c, 2);
  Serial.print("C. Humidity: ");
  Serial.print(humidity);
  Serial.println("%.");
}
//--------------------------------------------------------------------------------



Serial.println("CAL finished");
    myFile.write(SD0,4000);
    myFile.write(SD1,4000);
    myFile.write(TC,4);
    myFile.write(HC,4);
    myFile.close();
if(cal_rms==1)//for calibration of rms value and variation
  {
for(i=0;i<2000;i++)
{
avr_Vrms0 = avr_Vrms0+Vrms0[i];
avr_Vrms1 = avr_Vrms1+Vrms1[i];
}
avr_Vrms0 = avr_Vrms0/2000;
avr_Vrms1 = avr_Vrms1/2000;
Serial.println("RMS values");
Serial.println(avr_Vrms0,DEC);
Serial.println(avr_Vrms1,DEC);

for(i=0;i<2000;i++)
{
var_Vrms0=var_Vrms0+(Vrms0[i]-avr_Vrms0)*(Vrms0[i]-avr_Vrms0);
var_Vrms1=var_Vrms1+(Vrms1[i]-avr_Vrms1)*(Vrms1[i]-avr_Vrms1);
}
var_Vrms0=sqrt(var_Vrms0/3999);
var_Vrms1=sqrt(var_Vrms1/3999);
```

```
Serial.println("Variation values");
Serial.println(var_Vrms0,DEC);
Serial.println(var_Vrms1,DEC);
    }

Serial.println("Reading from SD");

  File dataFile = SD.open("testData.log");
  // if the file is available, write to it:
  if (dataFile) {
    while (dataFile.available()) {
      Serial.println(dataFile.read(),HEX);
    }
    dataFile.close();
  }
  // if the file isn't open, pop up an error:
  else {
    Serial.println("error opening testData.log");
       }

          while(1){
          digitalWrite(8,LOW);
          delay(500);
          digitalWrite(8,HIGH);
          delay(500);
        }
    }else{    //Monitoring mode
          if(ib==length){


            inc=CG(buf1_0, 0, inc);
            CG(buf1_1, 1, inc);
              for(j=0;j<size;j++)
              {
              vdc0=vdc0+buf1_0[j];
              vdc1=vdc1+buf1_1[j];
              }
              vdc0=vdc0/size;
              vdc1=vdc1/size;
              for(j=0;j<size;j++)
              {
              sum0 = sum0+(buf1_0[j]-vdc0)*w[j]*(buf1_0[j]-vdc0)*w[j];
              sum1 = sum1+(buf1_1[j]-vdc1)*w[j]*(buf1_1[j]-vdc1)*w[j];
              }
         for(j=0;j<300;j++)
              {
              buf1_0[j]=(buf1_0[j]-vdc0);
              }
                Vrms0[0]=1.581*3.3*sqrt(sum0/length)/4096;
                Vrms1[0]=1.581*3.3*sqrt(sum1/length)/4096;
                sum0 = 0;
                sum1 = 0;
                vdc0=0;
                vdc1=0;
         SD0int=Vrms0[0]*10;
         SD0frac=Vrms0[0]*1000-SD0int*100;
         SD1int=Vrms1[0]*10;
```

```
          SD1frac=Vrms1[0]*1000-SD1int*100;
          MSD0[SDidx++]=SD0int;
          MSD0[SDidx++]=SD0frac;
          MSD0[SDidx++]=SD1int;
          MSD0[SDidx++]=SD1frac;
          /*Serial.print("Vrms0= ");
            Serial.println(Vrms0[0]);
            Serial.print("Vrms1= ");
            Serial.println(Vrms1[0]);*/

  for(sft=20;sft<200;sft++)// the value of shift
   {
    for(auto_idx=0;sft+auto_idx<300;auto_idx++)// index for the buffer
     {
      buf_autoco_1[sft] =buf_autoco_1[sft]+buf1_0[auto_idx]*buf1_0[sft+auto_idx];
     }
     buf_autoco_p[sft]=buf_autoco_1[sft];
   }
j=0;
i=0;
for(sft=21;sft<199;sft++)
{
if(j<10)
   {
if(buf_autoco_1[sft]>buf_autoco_1[sft-1])
{
  if(buf_autoco_1[sft]>buf_autoco_1[sft+1])
  {
  maxauto[j]=buf_autoco_1[sft];
  maxautoidx[j] =sft;
  j++;
  }
}
   }
}

maxidx1=maxautoidx[0];
automax1=maxauto[0];
for(i=0;i<10;i++)
{
if(automax1<maxauto[i])
{
automax1=maxauto[i];
maxidx1=maxautoidx[i];
}
}

autoval00  = ((buf_autoco_1[maxidx1-1]>>24)&0x000000ff);
autoval01  = ((buf_autoco_1[maxidx1-1]>>16)&0x000000ff);
autoval10  = ((buf_autoco_1[maxidx1]>>24)&0x000000ff);
autoval11  = ((buf_autoco_1[maxidx1]>>16)&0x000000ff);
autoval20  = ((buf_autoco_1[maxidx1+1]>>24)&0x000000ff);
autoval21  = ((buf_autoco_1[maxidx1+1]>>16)&0x000000ff);

for(auto_idx=0;auto_idx<300;auto_idx++)
{
buf_autoco_1[auto_idx]=0;
```

```
}

MSD0[SDidx++]=maxidx1;
MSD0[SDidx++]=autoval00;
MSD0[SDidx++]=autoval01;
MSD0[SDidx++]=autoval10;
MSD0[SDidx++]=autoval11;
MSD0[SDidx++]=autoval20;
MSD0[SDidx]=autoval21;

myFile.write(MSD0,11);

SDidx=0;
        }
        else if(ib==2*length){
          inc=CG(buf2_0, 0, inc);
          CG(buf2_1, 1, inc);
             tstart=micros();
              for(j=0;j<size;j++)
              {
              vdc0=vdc0+buf2_0[j];
              vdc1=vdc1+buf2_1[j];
              }
              vdc0=vdc0/size;
              vdc1=vdc1/size;

              for(j=0;j<size;j++)
              {
              sum0 = sum0+(buf2_0[j]-vdc0)*w[j]*(buf2_0[j]-vdc0)*w[j];
              sum1 = sum1+(buf2_1[j]-vdc1)*w[j]*(buf2_1[j]-vdc1)*w[j];
              }
            for(j=0;j<300;j++)
              {
              buf2_0[j]=(buf2_0[j]-vdc0);
              }
                Vrms0[0]=1.581*3.3*sqrt(sum0/length)/4096;
                Vrms1[0]=1.581*3.3*sqrt(sum1/length)/4096;
                sum0 = 0;
                sum1 = 0;
               vdc0=0;
               vdc1=0;
         SD0int=Vrms0[0]*10;
         SD0frac=Vrms0[0]*1000-SD0int*100;
         SD1int=Vrms1[0]*10;
         SD1frac=Vrms1[0]*1000-SD1int*100;
         MSD1[SDidx++]=SD0int;
         MSD1[SDidx++]=SD0frac;
         MSD1[SDidx++]=SD1int;
         MSD1[SDidx++]=SD1frac;
        /*Serial.print("Vrms0= ");
          Serial.println(Vrms0[0]);
          Serial.print("Vrms1= ");
          Serial.println(Vrms1[0]);*/

for(sft=20;sft<200;sft++)// the value of shift
     {
for(auto_idx=0;sft+auto_idx<300;auto_idx++)// index for the buffer
```

```
 {
buf_autoco_2[sft] =buf_autoco_2[sft]+buf2_0[auto_idx]*buf2_0[sft+auto_idx];
    }
      }
j=0;
i=0;
for(sft=21;sft<199;sft++)
{
if(j<10)
   {
if(buf_autoco_2[sft]>buf_autoco_2[sft-1])
{
  if(buf_autoco_2[sft]>buf_autoco_2[sft+1])
  {
  maxauto1[j]=buf_autoco_2[sft];
  maxautoidx1[j]=sft;
  j++;
  }
}
   }
}
maxidx1=maxautoidx1[0];
automax2=maxauto1[0];
for(i=0;i<10;i++)
{
if(automax2<maxauto1[i])
{
automax2=maxauto1[i];
maxidx1=maxautoidx1[i];
}
}

autoval00  = ((buf_autoco_2[maxidx1-1]>>24)&0x000000ff);
autoval01  = ((buf_autoco_2[maxidx1-1]>>16)&0x000000ff);
autoval10  = ((buf_autoco_2[maxidx1]>>24)&0x000000ff);
autoval11  = ((buf_autoco_2[maxidx1]>>16)&0x000000ff);
autoval20  = ((buf_autoco_2[maxidx1+1]>>24)&0x000000ff);
autoval21  = ((buf_autoco_2[maxidx1+1]>>16)&0x000000ff);

for(auto_idx=0;auto_idx<300;auto_idx++)
{
buf_autoco_2[auto_idx]=0;
}



MSD1[SDidx++]=maxidx1;
MSD1[SDidx++]=autoval00;
MSD1[SDidx++]=autoval01;
MSD1[SDidx++]=autoval10;
MSD1[SDidx++]=autoval11;
MSD1[SDidx++]=autoval20;
MSD1[SDidx]=autoval21;
myFile.write(MSD1,11);

if ((count==500)||(count==1500)||(count==2500)){
  Serial.println(count);
  temp_c = sht1xTH.readTemperatureC();
```

```
  humidity = sht1xTH.readHumidity();
  tint=temp_c;
  tfloat=temp_c*100-tint*100;
  hint=humidity;
  hfloat=humidity*100-hint*100;
  if (count==500){
    t=1;}
  else if (count==1500){
    t=3;}
  else if (count ==2500){
    t=5;}

  T[t]=tint;
  T[t+1]=tfloat;

  H[t]=hint;
  H[t+1]=hfloat;

  Serial.print(" Temperature: ");
  Serial.print(temp_c, 2);
  Serial.print("C. Humidity: ");
  Serial.print(humidity);
  Serial.println("%.");
}
//--------------------------------------------------------------------------------------------
SDidx=0;
count++;
tstop=micros();
        }
if (count==1000){
    myFile.write(T,8);
    myFile.write(H,8);

    myFile.close();
    Serial.println(" ");
    Serial.println(" ");
    Serial.println(nameIndex);
    Serial.println(" ");
    Serial.println(" ");
            digitalWrite(9,HIGH);
            fileName = "L";
            fileName += nameIndex;                 // file name index
            fileName += ".log" ;                   // file extension
            fileName.toCharArray(bufferName,10) ;  // convert String to char
            // file name example : L6.log
            myFile=SD.open(bufferName,FILE_WRITE); // open the new file
            nameIndex++;                           // increase the index name
            count=1;                              // reset counter
            digitalWrite(9,LOW);
        }
    }
}
void SetPGA(int G, int Ch, int CS) {
 SPI.transfer(CS, 0x2A, SPI_CONTINUE);
 byte response= SPI.transfer(CS,(G<<4) + Ch);
 SPI.endTransaction();
}
```

```
int CG(int Buf[300], int ch, int inc) {
 Array<int> array = Array<int>(Buf,size);
          if (ch==0){
                Maxbuf0=array.getMax();
                Minbuf0=array.getMin();
                Vpp0=Maxbuf0-Minbuf0;
                  /*Serial.print("Minimo01= ");
                  Serial.println(Minbuf0);
                   Serial.print("Vpp01= ");
                  Serial.println(Vpp0);
                  Serial.print("Massimo01= ");
                  Serial.println(Maxbuf0);*/

          if ((Vpp0<(0.4*VT))&&(Minbuf0>1200)){
            inc=inc+1;
            if (inc==incmax){
                inc=0;
                if (Gan0[0]<7){

                  Gan0[0]++;
                  Gan0[1]=1;}
                  else if (Gan0[0]==7){
                    Gan0[1]=1;}}
                Serial.print("Gan01= ");
                  Serial.println(Gan0[0]);
                /*Serial.print("Massimo01= ");
                  Serial.println(Maxbuf0);*/} // =1 quiere decir que los valores salvados
                  son buenos
                  else if ((Vpp0>=(0.9*VT))||(Minbuf0<200)){
                    inc=0;
                  if (Gan0[0]>0){

                  Gan0[0]--;
                  Gan0[1]=0;}
                  else if (Gan0[0]==0){
                    Gan0[1]=0;}
                Serial.print("Gan02= ");
                  Serial.println(Gan0[0]);
                /*Serial.print("Massimo02= ");
                  Serial.println(Maxbuf0);*/}
                  else {
                    inc=0;
                  Gan0[1]=1;
                Serial.print("Gan03= ");
                  Serial.println(Gan0[0]);
               /* Serial.print("Massimo03= ");
                  Serial.println(Maxbuf0);*/}
                myFile.write(Gan0,3);

                SetPGA(Gan0[0],0,52);}
                else if (ch==1){
                  Maxbuf1=array.getMax();
                Minbuf1=array.getMin();
                Vpp1=Maxbuf1-Minbuf1;

          if ((Vpp1<(0.4*VT))&&(Minbuf0>1200)){
                if (Gan1[0]<7){
```

```
                  Gan1[0]++;
                  Gan1[1]=1;}
                  else if (Gan1[0]==7){
                    Gan1[1]=1;}
                Serial.print("Gan11= ");
                  Serial.println(Gan1[0]);
                /*Serial.print("Massimo11= ");
                  Serial.println(Maxbuf1);*/} // =1 quiere decir que los valores salvados
                  son buenos
                  else if ((Vpp1>=(0.9*VT))||(Minbuf0<100)){
                  if (Gan1[0]>0){

                  Gan1[0]--;
                  Gan1[1]=0;}
                  else if (Gan1[0]==0){
                    Gan1[1]=0;}
                Serial.print("Gan12= ");
                  Serial.println(Gan1[0]);
                /*Serial.print("Massimo12= ");
                  Serial.println(Maxbuf1);*/}
                  else {
                  Gan1[1]=1;
                Serial.print("Gan13= ");
                  Serial.println(Gan1[0]);
                /*Serial.print("Massimo13= ");
                  Serial.println(Maxbuf1);*/}
                myFile.write(Gan1,3);

                  SetPGA(Gan1[0],0,10);}
                 return inc;
}
```

8. Allegato 2

Codice del programma dei dati grezzi.

```
//-----------------------------------------------------------------------
#include <Array.h>    //Per ottenere il massimo, il minimo e Vpp
#include <SHT1xTH.h>
// Specify data and clock connections and instantiate SHT1x object
#define dataPin  11  // pins a cambiar porque ya estan utilizados para otras cosas en el
voice-care
#define clockPin 2  // pins a cambiar porque ya estan utilizados para otras cosas en el
voice-care
SHT1xTH sht1xTH(dataPin, clockPin);
//-----------------------------------------------------------------------
#include <SD.h>
#include <SPI.h>

const int size=300;
volatile uint16_t length=size;                 // buffer size
uint16_t ib = 0;                    // buffer index
uint32_t result;                   //value read from ADC

int buf1_0[size];
int buf1_1[size];
int buf2_0[size];
int buf2_1[size];
byte SD0[24000];
byte SD1[24000];
byte MSD0[2*size];
byte MSD1[2*size];
int SDidx=0;

int sft,auto_idx;
long ADCval;
long tstart;
long tstop;

char bufferName[10];
int count = 0;
int nameIndex = 1;                // index for saved file name
String fileName ="";
File myFile;
boolean calMode=0;
boolean cal_rms=1;
int chan=0;
int i,j,automax1,automax2, l;



float w[300];

float pi=3.141592653;

int rms_idx=0;
//-------------------------------------------------------------
float temp_c;
float humidity;

byte TC[4];
byte HC[4];
byte T[8];
```

```
byte H[8];

byte tint;
byte tfloat;
byte hint;
byte hfloat;
int t;

int Maxbuf0=0;
int Minbuf0=0;
int Maxbuf1=0;
int Minbuf1=0;
int Vpp0=0;
int Vpp1=0;
byte Gan0[3];
byte Gan1[3];
int VT=4096;
int inc=0;
int incmax=1;
//-------------------------------------------------------------


void setup() {
    Serial.begin(115200);
    pinMode(2,OUTPUT);    // port B pin 25
    analogWrite(2,255);   // sets up some other registers I haven't worked out yet
    pinMode(7,INPUT);
    pinMode(8,OUTPUT);
    pinMode(9,OUTPUT);
    digitalWrite(8,HIGH);

    SPI.begin(10);        //CS Amplificatore
    SPI.begin(52);
  Serial.println("Initializing SD card...");
    if (!SD.begin(4)){
      while(1){
        digitalWrite(9,LOW);
        digitalWrite(9,HIGH);
        digitalWrite(8,LOW);
        digitalWrite(9,HIGH);
    }
      }
for(i=0;i<300;i++)
{
 w[i] = 0.54 - 0.46*cos(2*pi*i/(length-1));
}
//-----------------------------
    Gan0[0]=6;
    Gan1[0]=6;
    Gan0[3]=0;
    Gan1[3]=1;
    SetPGA(Gan1[0],0,10);
    SetPGA(Gan0[0],0,52);

    TC[0]=0xFF;
    TC[3]=0xFF;
    HC[0]=0xFF;
```

```
    HC[3]=0xFF;
    T[0]=0xFF;
    T[7]=0xFF;
    H[0]=0xFF;
    H[7]=0xFF;
//-----------------------------
SD.remove("testData.log");
  if (SD.exists("testData.log")){
    Serial.println("testData.log exists.");
  }
  else {
    Serial.println("testData.log doesn't exist.");
  }

SD.remove("Monitor.log");
  if (SD.exists("Monitor.log")){
    Serial.println("Monitor.log exists.");
  }
  else {
    Serial.println("Monitor.log doesn't exist.");
  }

    if (!digitalRead(7)){
        calMode=0;
        digitalWrite(9,HIGH);
        Serial.println("Monitoring mode");
        myFile = SD.open("L0.log", FILE_WRITE);

    }else{
        calMode=1;
        digitalWrite(8,HIGH);
        Serial.println("CAL mode");
        myFile = SD.open("testData.log", FILE_WRITE);

    }
   setRegisters();
   Serial.println("set is ok");
}



void setRegisters()
{
 //WARNING
  //THE ARDUINO CHANNEL IS NOT THE SAME OF SAM PROCESSOR
  //ARDUINO SAM
  //AD7     CH0
  //AD0     CH7
  //AD6     CH1
  //AD1     CH6
  //AD2     CH5

      // Trigger Enable
      // Set TRGSEL TO 1: Timer Counter Channel 0 TIOA as trigger source
      // 12 bit resolution
      // normal mode
      // not use sequence registers
```

```
  REG_TC0_WPMR=0x54494D00; // enable write to registers of TC
  REG_TC0_CMR0=0b00000000000010011100010000000000; // set channel mode register (see
  datasheet)
  REG_TC0_RC0=2100; // counter period
  REG_TC0_RA0=1000;  // PWM value
  REG_TC0_CCR0=0b101;    // start counter
  REG_TC0_IER0=0b00010000; // enable interrupt on counter=rc
  REG_TC0_IDR0=0b11101111; // disable other interrupts
  NVIC_EnableIRQ(TC0_IRQn); // enable TC0 interrupts

  ADC->ADC_MR = 0x10310103;

  ADC->ADC_CHDR = 0xFFFF ;      // disable all channels
  adc_enable_channel(ADC, ADC_CHANNEL_7); //ADC A0 ON ARDUINO CORRESPOND CH7 ON SAM

  adc_enable_interrupt(ADC, ADC_IER_DRDY);

  //TIMER TRIGGER
  //adc_configure_trigger(ADC, ADC_TRIG_TIO_CH_0, 0);

  NVIC_EnableIRQ(ADC_IRQn);// enable ADC interrupts

  adc_start(ADC);
}


void TC0_Handler()
{
    long dummy=REG_TC0_SR0; // vital - reading this clears some flag
                          // otherwise you get infinite interrupts
}

void ADC_Handler(){
 if ((adc_get_status(ADC) & ADC_ISR_DRDY) == ADC_ISR_DRDY)
  {
        if (ib < length) {
          if(chan==0){
    result =adc_get_channel_value(ADC,ADC_CHANNEL_7);
            buf1_0[ib]= result;
            chan=1;
            ADC->ADC_CHDR = 0xFFFF ;      // disable all channels
            adc_enable_channel(ADC, ADC_CHANNEL_6);
                     }
         else if(chan==1){
    result =adc_get_channel_value(ADC,ADC_CHANNEL_6);
            buf1_1[ib++]= result;
            chan=0;
            ADC->ADC_CHDR = 0xFFFF ;      // disable all channels
            adc_enable_channel(ADC, ADC_CHANNEL_7);
        }
        }
        else if (ib >= length && ib < 2*length) {
          if(chan==0){
    result =adc_get_channel_value(ADC,ADC_CHANNEL_7);
            buf2_0[ib - length]= result;
            chan=1;
            ADC->ADC_CHDR = 0xFFFF ;      // disable all channels
```

```
            adc_enable_channel(ADC, ADC_CHANNEL_6);
          }
          else if(chan==1){
    result =adc_get_channel_value(ADC,ADC_CHANNEL_6);
            buf2_1[ib++ - length]= result;
            chan=0;
            ADC->ADC_CHDR = 0xFFFF ;      // disable all channels
            adc_enable_channel(ADC, ADC_CHANNEL_7);
          }
        } else if (ib==2*length) {
          if(chan==0){
    result =adc_get_channel_value(ADC,ADC_CHANNEL_7);
            buf1_0[0] = result;
            chan=1;
            ADC->ADC_CHDR = 0xFFFF ;      // disable all channels
            adc_enable_channel(ADC, ADC_CHANNEL_6);
              }
         else if(chan==1){
    result =adc_get_channel_value(ADC,ADC_CHANNEL_6);
            buf1_1[0] = result;
            chan=0;
            ADC->ADC_CHDR = 0xFFFF ;      // disable all channels
            adc_enable_channel(ADC, ADC_CHANNEL_7);
            ib=1;
           }
        }
}
    long ADCval = REG_ADC_LCDR;//vital to return to void loop!
}

void loop(){
if (calMode==1){
while (count<20){
if(ib==length){  //buffer 1
              for(j=0;j<size;j++)
              {
                SD0[SDidx]=(buf1_0[j]>>8)&0x000f;// right shift 8 bits to keep the higher 4
                bits
                SD1[SDidx++]=(buf1_1[j]>>8)&0x000f;// right shift 8 bits to keep the higher
                4 bits
                SD0[SDidx]=(buf1_0[j])&0x00ff;// keep the lower 8 bits
                SD1[SDidx++]=(buf1_1[j])&0x00ff;// keep the lower 8 bits
              }
              inc=CG(buf1_0, 0, inc);
              CG(buf1_1, 1, inc);

//tstop=micros();
            }

else if(ib==2*length){//buffer 2
              for(j=0;j<size;j++)
              {
                SD0[SDidx]=(buf2_0[j]>>8)&0x000f;// right shift 8 bits to keep the higher 4
                bits
                SD1[SDidx++]=(buf2_1[j]>>8)&0x000f;// right shift 8 bits to keep the higher
                4 bits
                SD0[SDidx]=(buf2_0[j])&0x00ff;// keep the lower 8 bits
```

```
              SD1[SDidx++]=(buf2_1[j])&0x00ff;// keep the lower 8 bits
            }
            inc=CG(buf2_0, 0, inc);
            CG(buf2_1, 1, inc);
              count++;
              Serial.println(count);
          }
  }


adc_stop(ADC);
//--------------------------------------------------------------------------------
if (count==1000){
  Serial.println(count);
  temp_c = sht1xTH.readTemperatureC();
  humidity = sht1xTH.readHumidity();
  tint=temp_c;
  tfloat=temp_c*100-tint*100;
  TC[1]=tint;
  TC[2]=tfloat;
  hint=humidity;
  hfloat=humidity*100-hint*100;
  HC[1]=hint;
  HC[2]=hfloat;
  Serial.print(" Temperature: ");
  Serial.print(temp_c, 2);
  Serial.print("C. Humidity: ");
  Serial.print(humidity);
  Serial.println("%.");
}
//--------------------------------------------------------------------------------


Serial.println("CAL finished");
    myFile.write(SD0,24000);
    myFile.write(SD1,24000);
    myFile.write(TC,4);
    myFile.write(HC,4);
    myFile.close();

    Serial.println("CAL finished, dati salvati & file closed");
      while(1){
          digitalWrite(8,LOW);
          delay(500);
          digitalWrite(8,HIGH);
          delay(500);
        }


    }else if(calMode==0){    //Monitoring mode
          if(ib==length){
              for(j=0;j<size;j++)
              {
                MSD0[SDidx]=(buf1_0[j]>>8)&0x000f;// right shift 8 bits to keep the higher 4
                bits
                MSD1[SDidx++]=(buf1_1[j]>>8)&0x000f;// right shift 8 bits to keep the higher
                4 bits
```

```
              MSD0[SDidx]=(buf1_0[j])&0x00ff;// keep the lower 8 bits
              MSD1[SDidx++]=(buf1_1[j])&0x00ff;// keep the lower 8 bits
            }

    //-------------------------------------------------------------------------------------------
    -----------------
    myFile.write(MSD0,600);
    myFile.write(MSD1,600);

    inc=CG(buf1_0, 0, inc);
    CG(buf1_1, 1, inc);

    //-------------------------------------------------------------------------------------------
    -----------------
    SDidx=0;

        }
        else if(ib==2*length){
          tstart=micros();
          for(j=0;j<size;j++)
              {
                MSD0[SDidx]=(buf1_0[j]>>8)&0x000f;// right shift 8 bits to keep the higher 4
                bits
                MSD1[SDidx++]=(buf1_1[j]>>8)&0x000f;// right shift 8 bits to keep the higher
                4 bits
                MSD0[SDidx]=(buf1_0[j])&0x00ff;// keep the lower 8 bits
                MSD1[SDidx++]=(buf1_1[j])&0x00ff;// keep the lower 8 bits
              }
         myFile.write(MSD0,600);
    myFile.write(MSD1,600);

              inc=CG(buf2_0, 0, inc);
              CG(buf2_1, 1, inc);

if ((count==500)||(count==1500)||(count==2500)){
  Serial.println(count);
  temp_c = sht1xTH.readTemperatureC();
  humidity = sht1xTH.readHumidity();
  tint=temp_c;
  tfloat=temp_c*100-tint*100;
  hint=humidity;
  hfloat=humidity*100-hint*100;
  if (count==500){
    t=1;}
  else if (count==1500){
    t=3;}
  else if (count ==2500){
    t=5;}

  T[t]=tint;
  T[t+1]=tfloat;

  H[t]=hint;
  H[t+1]=hfloat;

  Serial.print(" Temperature: ");
  Serial.print(temp_c, 2);
```

```
  Serial.print("C. Humidity: ");
  Serial.print(humidity);
  Serial.println("%.");
}
    SDidx=0;
count++;
Serial.print(" count: ");
  Serial.print(count);
tstop=micros();

        }
if (count==1000){
    myFile.write(T,8);
    myFile.write(H,8);
    myFile.close();
    Serial.println("nameIndex");
    Serial.println(nameIndex);
            digitalWrite(9,HIGH);

            fileName = "L";
            fileName += nameIndex;
            fileName += ".log" ;
            fileName.toCharArray(bufferName,10) ;
            // file name example : L6.log
            myFile=SD.open(bufferName,FILE_WRITE); // open the new file
            nameIndex++;                            // increase the index name
            count=1;                              // reset counter
            digitalWrite(9,LOW);
        }
    }
}
void SetPGA(int G, int Ch, int CS) {
 SPI.transfer(CS, 0x2A, SPI_CONTINUE);
 byte response= SPI.transfer(CS,(G<<4) + Ch);
 SPI.endTransaction();
}
int CG(int Buf[300], int ch, int inc) {
 Array<int> array = Array<int>(Buf,size);
          if (ch==0){
                Maxbuf0=array.getMax();
                Minbuf0=array.getMin();
                Vpp0=Maxbuf0-Minbuf0;
                Serial.print("Minimo01= ");
                  Serial.println(Minbuf0);
                   Serial.print("Vpp01= ");
                  Serial.println(Vpp0);

          if ((Vpp0<(0.4*VT))&&(Minbuf0>1200)/*||(Maxbuf<(0.4*VT))*/){
            inc=inc+1;
            Serial.print(" inc: ");
            Serial.print(inc);
            if (inc==incmax){
                inc=0;
                if (Gan0[0]<7){

                  Gan0[0]++;
                  Gan0[1]=1;}
```

```
          else if (Gan0[0]==7){
            Gan0[1]=1;}}
        Serial.print("Gan01= ");
          Serial.println(Gan0[0]);
        Serial.print("Massimo01= ");
          Serial.println(Maxbuf0);} // =1 quiere decir que los valores salvados son
          buenos
          else if ((Vpp0>=(0.9*VT))||(Minbuf0<200)/*||(Maxbuf>=(0.95*VT))*/){
            inc=0;
          if (Gan0[0]>0){

          Gan0[0]--;
          Gan0[1]=0;}
          else if (Gan0[0]==0){
            Gan0[1]=0;}
        Serial.print("Gan02= ");
          Serial.println(Gan0[0]);
        Serial.print("Massimo02= ");
          Serial.println(Maxbuf0);}
          else {
            inc=0;
          Gan0[1]=1;
        Serial.print("Gan03= ");
          Serial.println(Gan0[0]);
        Serial.print("Massimo03= ");
          Serial.println(Maxbuf0);}
        myFile.write(Gan0,3);

        SetPGA(Gan0[0],0,52);}
        else if (ch==1){
          Maxbuf1=array.getMax();
        Minbuf1=array.getMin();
        Vpp1=Maxbuf1-Minbuf1;

    if ((Vpp1<(0.4*VT))&&(Minbuf0>1200)/*||(Maxbuf<(0.4*VT))*/){
        if (Gan1[0]<7){

          Gan1[0]++;
          Gan1[1]=1;}
          else if (Gan1[0]==7){
            Gan1[1]=1;}
        Serial.print("Gan11= ");
          Serial.println(Gan1[0]);
        Serial.print("Massimo11= ");
          Serial.println(Maxbuf1);} // =1 quiere decir que los valores salvados son
          buenos
          else if ((Vpp1>=(0.9*VT))||(Minbuf0<200)/*||(Maxbuf>=(0.95*VT))*/){
          if (Gan1[0]>0){

          Gan1[0]--;
          Gan1[1]=0;}
          else if (Gan1[0]==0){
            Gan1[1]=0;}
        Serial.print("Gan12= ");
          Serial.println(Gan1[0]);
        Serial.print("Massimo12= ");
          Serial.println(Maxbuf1);}
```

```
                  else {
                  Gan1[1]=1;
                Serial.print("Gan13= ");
                  Serial.println(Gan1[0]);
                Serial.print("Massimo13= ");
                  Serial.println(Maxbuf1);}
                myFile.write(Gan1,3);

                  SetPGA(Gan1[0],0,10);}

                  return inc;
}
```

RESUMEN

En este documento se presenta la continuación del desarrollo de un dispositivo basado en Arduino para la monitorización de la voz. Utiliza dos micrófonos, uno de aire y otro de contacto o garganta conectados como canales de entrada a la placa Arduino. También tiene otros dos componentes: un sensor de temperatura y humedad para la caracterización del entorno en el que se realiza la monitorización y un par de amplificadores programables para aumentar la resolución de las señales de entrada.

El objetivo principal del dispositivo es la caracterización de la voz mediante el cálculo de los parámetros básicos. El objetivo secundario es que estos parámetros pueden servir a estudios futuros en el área clínica como: tratar de identificar los riesgos de sufrir problemas en las cuerdas vocales, monitorizar personas con problemas en las cuerdas vocales y sin ellos para encontrar una relación entre hábitos y trastornos... Por esta razón, es un dispositivo destinado a ser utilizado durante todo el día, especialmente por las personas que trabajan mucho con la voz como profesores y cantantes. Esta posibilidad de monitorización fuera del entorno clínico y durante periodos de tiempo prolongados ofrece datos más reales.

El funcionamiento del dispositivo consiste en la monitorización de la voz a través de muestreo continuo de las dos señales y el cálculo de los parámetros en tiempo real, caracterizar el medio ambiente (ruido, temperatura y humedad), y el análisis de nuevo más en profundidad con la información almacenada en la tarjeta SD.

Tiene dos modos de funcionamiento, uno de calibración y otro de seguimiento propiamente. El primero se utiliza para calibrar los micrófonos y verificar el funcionamiento correcto después de conectar todo, y el segundo para realizar la monitorización.

El trabajo realizado consiste en la implementación del dispositivo, la realización test parciales y totales para verificar su correcto funcionamiento, y una prueba de monitorización que es el objetivo del trabajo.

3. TRABAJO DESARROLLADO:

En este capítulo, se expone el trabajo desarrollado, los componentes del dispositivo y cómo funciona. Es una nueva versión de Voice - Care, dispositivo desarrollado en los últimos años por el Departamento de Electrónica y Telecomunicaciones (DET) del Politecnico di Torino.

3.1 Introducción:

Como se ha comentado anteriormente, el propósito de este trabajo es desarrollar un dispositivo con autonomía para un día entero de trabajo de forma continua. Se trata de un dispositivo similar a los explicados en el estado del arte. El objetivo es registrar de forma continua la voz y tomar conjuntos de muestras para calcular los parámetros elegidos para analizar las características de la voz y del ambiente en el que se realiza la prueba.

El dispositivo está formado por:

• Arduino due (Figura 9)

• Micrófono de contacto o garganta (Figura 10)

• Micrófono de aire (Figura 12)

• Sensor de temperatura y humedad Sensirion (Figura 13)

• Amplificadores programables (Figura 11)

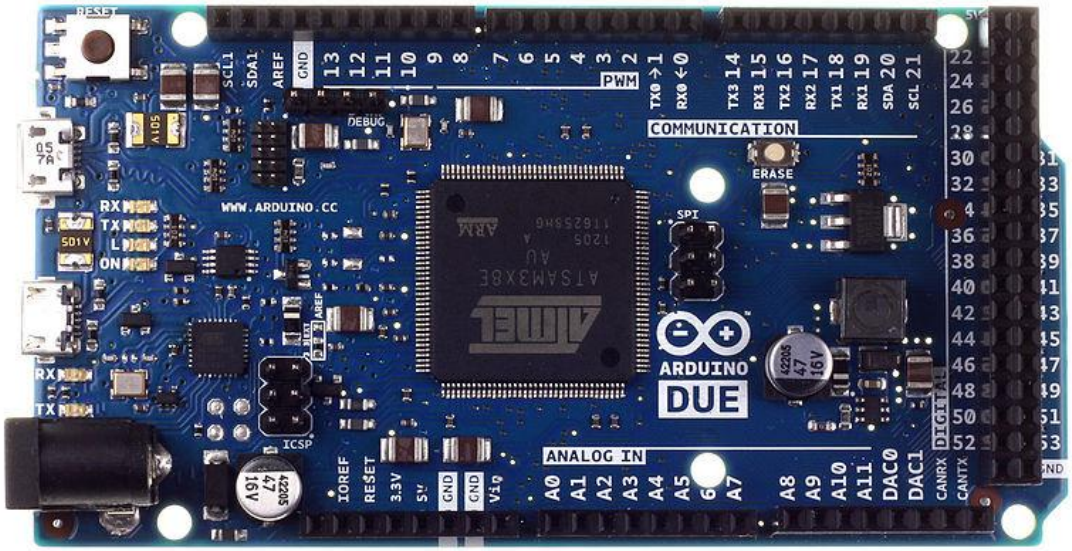


Figura 1: Arduino due

Figura 2: Microfono de garganta

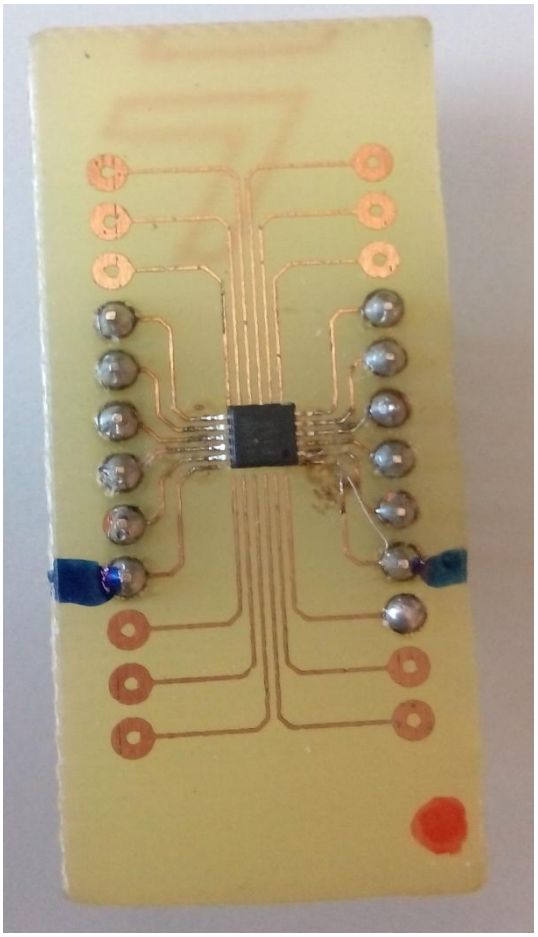


Figura 3: Amplificador programable

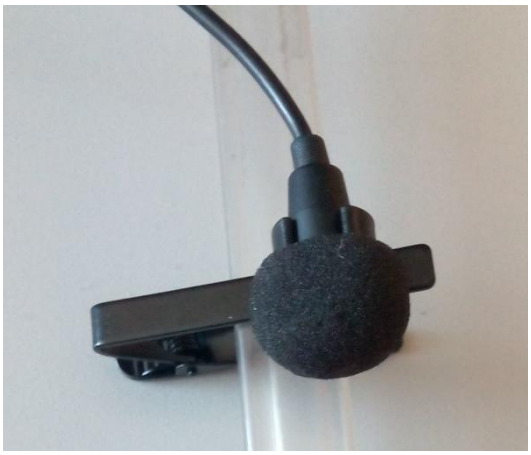


Figura 4: Microfono de aire

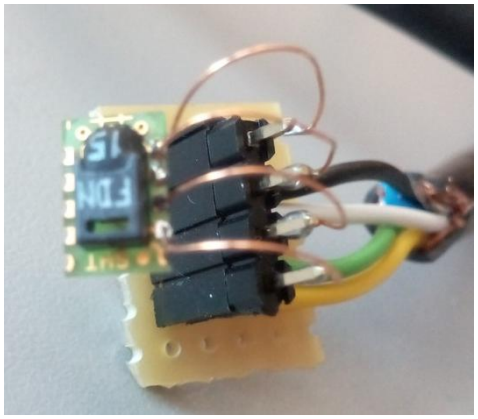


Figura 5: Sensirion

La registración se hace mediante un acelerómetro o micrófono de contacto, sujeto a la garganta con un una cinta médica (lo que atenúa menos las vibraciones). Con este micrófono toma la señal de voz antes de la modulación de la cavidad oral para calcular el F0 y la Vrms y el valor de la cantidad de tiempo real de habla respecto al tiempo total de grabación y así tener el Dt calculado para cada conjunto de datos en la elaboración posterior de los datos con Matlab. Así se puede tener una caracterización de la voz y también tienen la oportunidad de hacer un análisis posterior para identificar el riesgo de padecer trastornos de las cuerdas vocales.

Para conseguir una información más completa de las condiciones en las que se han tomado los datos y así poder analizar las posibles causas de una intensidad de voz mayor de lo normal para una persona, por ejemplo, se conecta un micrófono de aire cuyo trabajo consiste en tomar datos del ruido de fondo en el modo de monitorización y ayudaren en el modo de calibración. Para este mismo fin también ha añadido un sensor de temperatura y la humedad y tener así una caracterización del entorno en el que se toman los datos.

Además, la señal tomada a los micrófonos es muy baja. Para mejorar la calidad de los datos se ha añadido una etapa de amplificación programable entre los micrófonos y el microprocesador utilizado.

3.2. Descripxión general:

3.2.1 Dispositivo inicial:

Como se explica en la introducción se introdujo alguna mejora con respecto a la primera versión del dispositivo (VC) desarrollado para el departamento. Esta primera versión era un dispositivo portátil que adquiría datos brutos, las muestras de la señal de voz y las guardaba en una tarjeta SD. Los datos eran procesados para obtener los parámetros (nivel de presión sonora SPL, frecuencia fundamental F0 y tiempo de fonación Dt) necesarias para detectar y prevenir comportamientos vocales perjudiciales y los trastornos causados por éstos.

El VC tiene una componente hardware y una software cargado en el microcontrolador. El dispositivo puede funcionar de dos modos diferentes a elegir, cada uno con su propio propósito, calibración y monitorización.

En la calibración tenía dos canales de entrada, uno para el micrófono de contacto que mide el nivel de aceleración de sonido (SAL) a través de las vibraciones de la piel en el cuello y el otro para el micrófono del aire que mide el nivel de presión sonora (SPL) utilizado para la calibración y para transformar SAL en SPL que es la unidad de medida de la intensidad del sonido. En este modo de operación guarda sólo los datos de un pequeño periodo de tiempo.

La monitorización sólo trabajaba con uno de los canales, el que tiene el acelerómetro conectado para hacer la grabación de la voz de los datos brutos de la señal. En este modo de operación guarda los datos de forma continua guardando un archivo cada 3 minutos hasta que se apague o se detiene.

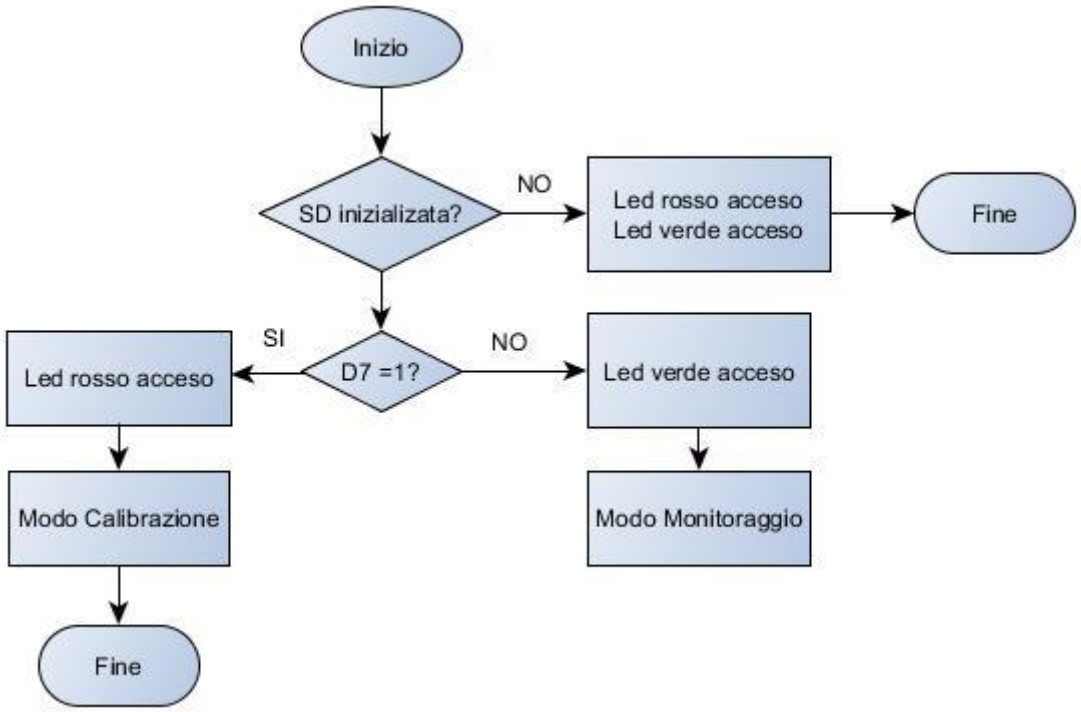


Figura 6: Diagramma di flusso primo VC

El proceso sigue el diagrama de flujo mostrado en la Figura 14. Las muestras son tomadas con una velocidad de muestreo de 38460 muestras por segundo en la calibración y 19230 en el seguimiento. Las muestras son analógicas por lo que se toman con el convertidor analógico - digital (ADC) de la tarjeta. Cada conjunto de datos tiene una duración de 30 ms.

En este primer prototipo se encontraron algunos problemas, tales como tramas de datos perdidas debido al reloj del microcontrolador Arduino Uno (Atmel ATmega328), es decir , no era lo suficientemente rápido como para guardar todos los datos en la SD en el tiempo disponible. Como no conseguía hacer los cálculos se hizo simplemente un dispositivo para la toma de muestras y no todas por las pérdidas comentadas.

3.2.2 Primera mejora

En ésta primera mejora del dispositivo se hicieron algunos cambios. El cambio más importante el de la tarjeta por otro modelo de Arduino más potente, el Arduino due que tiene otro microcontrolador (ARM SAM-3X8E), y el aumento en el número de canales de entrada. Al igual que en el primer prototipo, se utiliza la técnica de doble búfer, es decir, mientras se llena a uno con las muestras, se trabaja en el otro.

Mientras que la versión anterior no conseguía hacer los cálculos en el intervalo de tiempo en que llenaba uno de los bufers, ni a guardar todos los datos en SD, con este otro controlador consigue obtener los parámetros. Gracias a esto, los datos a guardar son muchos menos, un valor de cada parámetro por cada conjunto en lugar de todos los valores de las muestras y así es capaz de calcular los parámetros y guardarlos en SD.

Se resuelve de esta forma los dos principales problemas que tenía: No se pierde información porque guarda los datos de todos los conjuntos y no es sólo un dispositivo de muestreo, sino también de análisis de la señal tomada, asegurando la integridad del funcionamiento continuo. La nueva tarjeta también tiene mejoras en el ADC con una resolución más alta, 12 bits en lugar de 8.

En esta nueva versión del dispositivo, además de resolver problemas, añade nuevas funcionalidades.

- Se controla mejor la frecuencia de muestreo, de forma que en lugar de ser un múltiplo de los valores dados para el Arduino, se puede elegir la frecuencia deseada con el reloj. La frecuencia de muestreo en el modo de calibración se establece en 30.000 muestras por segundo y la de seguimiento para 20.000, que es de 30 kHz y 20 kHz.
- Tanto en el modo de calibración como en la monitorización, se añade otro canal de entrada para obtener información también del ruido de fondo por lo que, en esta versión hay tres canales en la calibración y dos en la monitorización.

En esta versión se estudia el mejor método (siempre hablando del presente caso) para hacer el cálculo de la frecuencia fundamental. Se estudia el método de la FFT y la correlación y, finalmente, se elege la correlación por la diferencia de carga computacional para el microcontrolador, para asegurar como se explicaba antes, el muestreo continuo durante el uso del dispositivo y la integridad de los valores guardados.

3.2.3 Ultima mejora

Finalmente se tiene la versión más actual del dispositivo que se lleva a cabo en este proyecto. Se continúa con la misma tarjeta y, por tanto, con el mismo microcontrolador (ARM- 3X8E SAM) , pero se hacen cambios a los canales y funcionalidades.

- Se quita el tercer canal para el modo de calibración, dejando sólo dos canales en ambos modos.
- Se añade el sensor de temperatura y humedad (Sensirion SHT15) para tener una caracterización más completa del ambiente. Este sensor se comunica con el Arduino por dos pines digitales.
- También se incluyen dos amplificadores programables, uno para cada canal de entrada, es decir, para cada micrófono conectado.

Todas estas mejoras, y la implementación de todo el dispositivo se explica en las siguientes secciones.

3.3 Implemetación:

Como ya se ha explicado, es un dispositivo cuya finalidad principal es tomar muestras de la señal de voz, calcular y guardar los parámetros especificados de forma continua durante todo el tiempo de operación. Además de este objetivo principal se han añadido otras funcionalidades.

Aunque el objetivo es conseguir los valores de los parámetros que se utilizan para caracterizar la voz y ser capaz de conocer el riesgo de trastornos de la voz mediante el análisis de estos parámetros, a veces es necesario para ver la señal completa, los datos brutos, no sólo para comprobar los cálculos o el funcionamiento de la funcionalidad del dispositivo, también para tener las grabaciones para análisis y estudios futuros tal vez con otra meta o enfoque.

El análisis de la señal de voz se realiza con las muestras tomadas por el micrófono de contacto. Las muestras tomadas con este micrófono son valores de SAL como ya se ha comentado, y esta es la razón del modo de funcionamiento de calibración, para encontrar la relación entre los valores de SAL y los de SPL. Una vez que haya encontrado esta información, se puede iniciar la grabación, el cálculo y guardar parámetros de forma continua en el tiempo.

Así, el dispositivo cuenta con dos modos de funcionamiento de selección manual: Calibración y Monitorización. Esta elección se realiza cambiando el valor de la entrada a un pin digital (D7). Como ambos modos necesitan comunicar con la tarjeta SD para guardar los valores, el primer paso es inicializar dicha tarjeta y comprobarlo. Si la tarjeta no inicializa de forma correcta, muestra un mensaje en la pantalla. Una vez hecho esto, se elige entre un modo u otro.

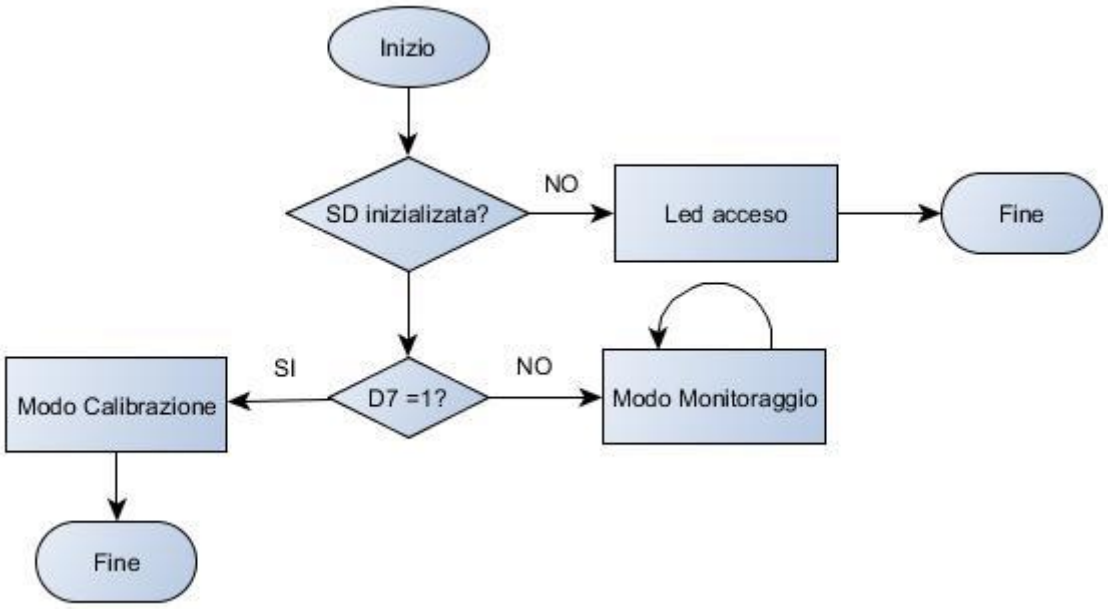


Figura 7: Diagramma di flusso generale

Una vez elegido el modo de monitorización, se establecen los valores de los registros de Arduino necesarios para la operación continua. Se elige el temporizador,se habilitan las interrupciones (pero sólo los de este temporizador, el resto se deshabilitan, no se habilitan) y establece un modo de funcionamiento del temporizador "free running", es decir, operación continua, cada vez que salta la interrupción, vuelve de nuevo al valor inicial. También se establece esta forma de la frecuencia de muestreo. En lugar de establecer esa frecuencia como un múltiplo o divisor la frecuencia de reloj de la

tarjeta, el valor se establece en el contador de tiempo y cada vez que se llega a este valor, salta una interrupción. Después, en la rutina de atención a la interrupción se establecen canales de muestreo, se toman los valores y todo. La frecuencia de muestreo ajustado a 10000 muestras por segundo. También se habilitan los canales de entrada para micrófonos, bloqueando el resto, y se inicia el ADC.

Puestos a punto todos los registros necesarios para su correcto funcionamiento, se debe establecer la rutina de atención a la interrupción. Es en esta rutina en donde se implementa la técnica del doble búfer. Al inicio del programa se ha declarado dos buffers de datos 300 para cada canal (30 ms muestreados a 10000 muestras por segundo). Cada vez que salta la interrupción, se toma una muestra de un canal del ADC; se bloquea ese canal, se habilita al segundo y se toma una muestra de otro canal, se bloquea de nuevo y se habilita el primero de modo que se queda a la espera de una nueva interrupción. El índice de posición del buffer se incrementa en cada interrupción y cuando se llena uno, se empieza a llenar el segundo (en realidad dos buffers porque son dos canales). En el tiempo que se tarda en llenar uno, el otro debe ser capaz de hacer todos los cálculos y guardar los datos para asegurar la operación continua.

3.3.1 SPI

La comunicación con la tarjeta SD y la ganancia de los amplificadores programables se hace con la interfaz periférica serie (SPI). Es un protocolo de datos en serie síncrona utilizada por microcontroladores para comunicarse con periféricos a corta distancia de forma rápida. Los componentes son:

- Dispositivo maestro: En este caso y generalmente, el microcontrolador.
- MISO (Master In Slave Out): Enviar datos del periférico al microcontrolador.
- MOSI (Master Out Slave In): Enviar datos del microcontrolador al periferico.
- SCL (Reloj serie) del reloj para sincronizar la comunicación.
- SS (Slave Select): Se refiere a la clavija que permite la comunicación con cualquier periférico SPI. En otros modelos Arduino se puede conectar a uno solo con esta interfaz, pero en el caso del due, se pueden conectar 3. Se utiliza uno para la SD y los otros dos para los dos amplificadores programables. En caso de tener más de un periférico, comparten MISO, MOSI y CLK, todo menos el SS.

Por tanto, se debe iniciar la interfaz SPI para cada periférico con los diferentes pines del SS. Utiliza el pin 4 para la SD y 10 y 52 para los amplificadores. El 52 para el canal del micrófono de garganta y el 10 para el micrófono de aire.

3.3.2 Ventana de peso

La ventana de peso es un vector de las mismas dimensiones del buffer, lleno con la función:

$$w[i] = 0{,}54 - 0{,}46 \cdot \cos\left(\frac{2 \cdot \pi \cdot i}{length - 1}\right)$$

donde i hace referencia a cada posición del vector y length a la dimensión del vector, es decir, 300 en este caso.

La función de esta ventana de pesaje es aliviar los errores causados por:

- tener una señal discreta debido al muestreo
- la trama de datos elegida para el análisis no es un múltiplo exacto del periodo de la señal.

Fijados todos los valores iniciales y la rutina de atención a la interrupción, se puede comenzar con el programa y sus modos.

3.3.3 Programa de parámetros

En el programa de datos elaborados además del muestreo de los datos, se calculan los parámetros salvando solo estos datos.

Calibración

La calibración como ya se ha comentado, tiene su razón de ser en la necesidad de encontrar una relación entre los valores de SAL tomados por el micrófono a contacto o acelerómetro y los valores SPL tomados por el micrófono que tiene las unidades de medida utilizadas para el sonido.

Para encontrar esta relación, primero debe calcular la constante del micrófono de aire para traducir los valores de Vrms calculados con las muestras tomadas por este canal y DB (unidades en que se mide el SPL) .

$SPL = 20log\left(\frac{P}{P_0}\right)$ donde $P_0 = 2 \cdot 10^5$ y $P = K_{mic} \cdot Vrms_{mic}$

Cada vez que el buffer está lleno, se hacen los calculos para obtener el valor de Vrms y hacer el control de ganancia.

Control de ganancia

El señal de hablado normal del ser humano, es decir, sin gritar o susurrar, es muy bajo respecto al fondo de escala disponible. Aunque los parámetros necesarios se pueden calcular igualmente sin este paso, cuanta mayor sensibilidad tenga el muestreo, más exactos serán los parámetros y más información se podrá obtener del análisis.

El amplificador utilizado es el PGA112. Es un amplificador programable con comunicación SPI con Arduino. Este modelo tiene 8 niveles de ganancia binarios, es decir, para cada nivel de ganancia más, la ganancia real se incrementa al doble.

Tabella 1: Tabella dei guadagni

Ganancia en binario	Ganancia real
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128

Tiene diferentes modos de funcionamiento en función de la funcionalidad elegida. Para el proyecto se ha elegido para trabajar con un canal de entrada y un Vref = 1.65V. Funciona sólo con un canal de entrada, incluso si se tiene la oportunidad de conectar dos canales porque la diferencia entre las dos señales (micrófono de contacto y distancia micrófono) necesita un control diferenciado de la ganancia si se quiere tener la mayor sensibilidad para cada uno. La razón de la añadir una componente de continua a la señal de la mitad del valor del fondo de escala del ADC es transformar la señal bipolar de los micrófonos en señales unipolares por las características del ADC que no permite la entrada de valores negativos. De esta forma, la señal centrada en 0 originalmente, sube a 1,65 donde está centrada ahora. La lógica de funcionamiento se muestra en el siguiente diagrama de flujo:

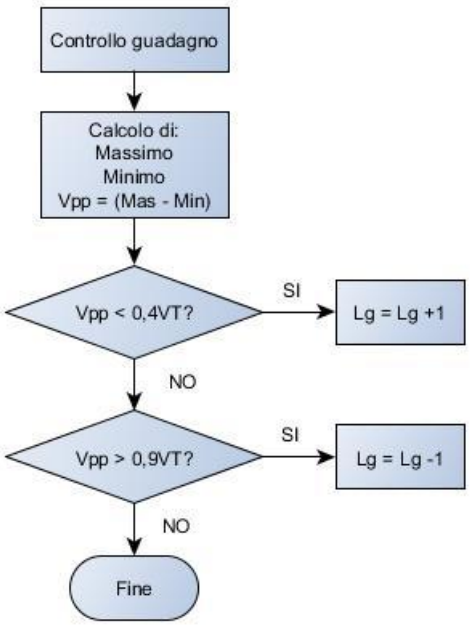


Figura 8: Diagramma de flusso: Controllo di guadagno

Es decir, se calcula el valor máximo y mínimo, y la diferencia entre ellos, del vector de muestras brutas y se comparan estos valores con el fondo de escala encontrando tres situaciones diferentes:

- Si la diferencia entre ellos es inferior al 40% del fondo de escala, hay que aumentar el nivel de ganancia.
- Si la diferencia es superior al 90% del fondo de escala, reduce el nivel de ganancia.
- Si la diferencia está entre 40% y 90%, no hacer ningún cambio.

Para aumentar la seguridad de evitar la saturación, tanto superior como inferior, se añaden límites a las situaciones anteriores y para evitar problemas de llegar por software a valores de ganancia inexistentes, también se agregan las condiciones de no se elevar por encima del nivel 7 ni bajar del nivel 0.

Como se puede ver a continuación, en el experimento 6, al probar el algoritmo con la señal real de voz, satura demasiado, se deben descartar demasiadas tramas del canal del micrófono de contacto. Por esta razón, se añade un retardo en el aumento de ganancia del micrófono de garganta, en el micrófono de aire no es necesario hacer el cambio. Después de las mejoras, las situaciones y condiciones (por el canal sin retardo) son:

- Si la diferencia entre ellos es inferior al 40% del fondo de escala, el valor máximo de muestreo es menor de 1200 y el valor de la ganancia actual es menor de 7, se aumenta la ganancia.
- Si la diferencia es mayor que 90% del fondo de escala, el valor mínimo de muestreo es mayor que 100 y el valor de ganancia actual es mayor que 0, disminuye el nivel de ganancia.
- Si la diferencia está entre 40% y 90%, no hacer ningún cambio.

El algoritmo final para el canal del micrófono de contacto con el aumento de ganancia retardado se muestra en la siguiente figura:

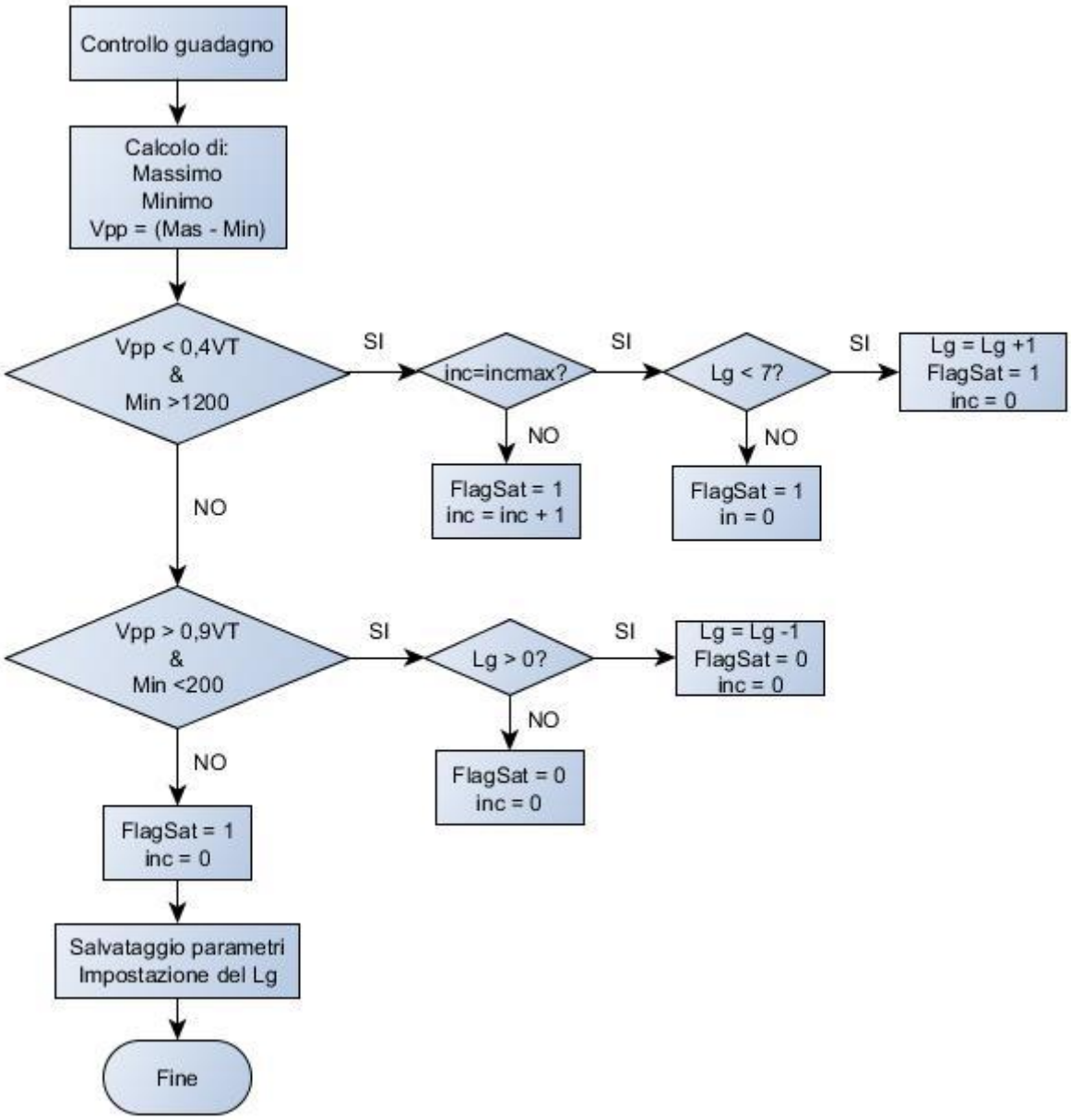


Figura 9: Controllo guadagno con incmax

El valor del parámetro incmax se puede cambiar según convenga.

El algoritmo final para el canal del micrófono de aire sin retardo en el aumento es:

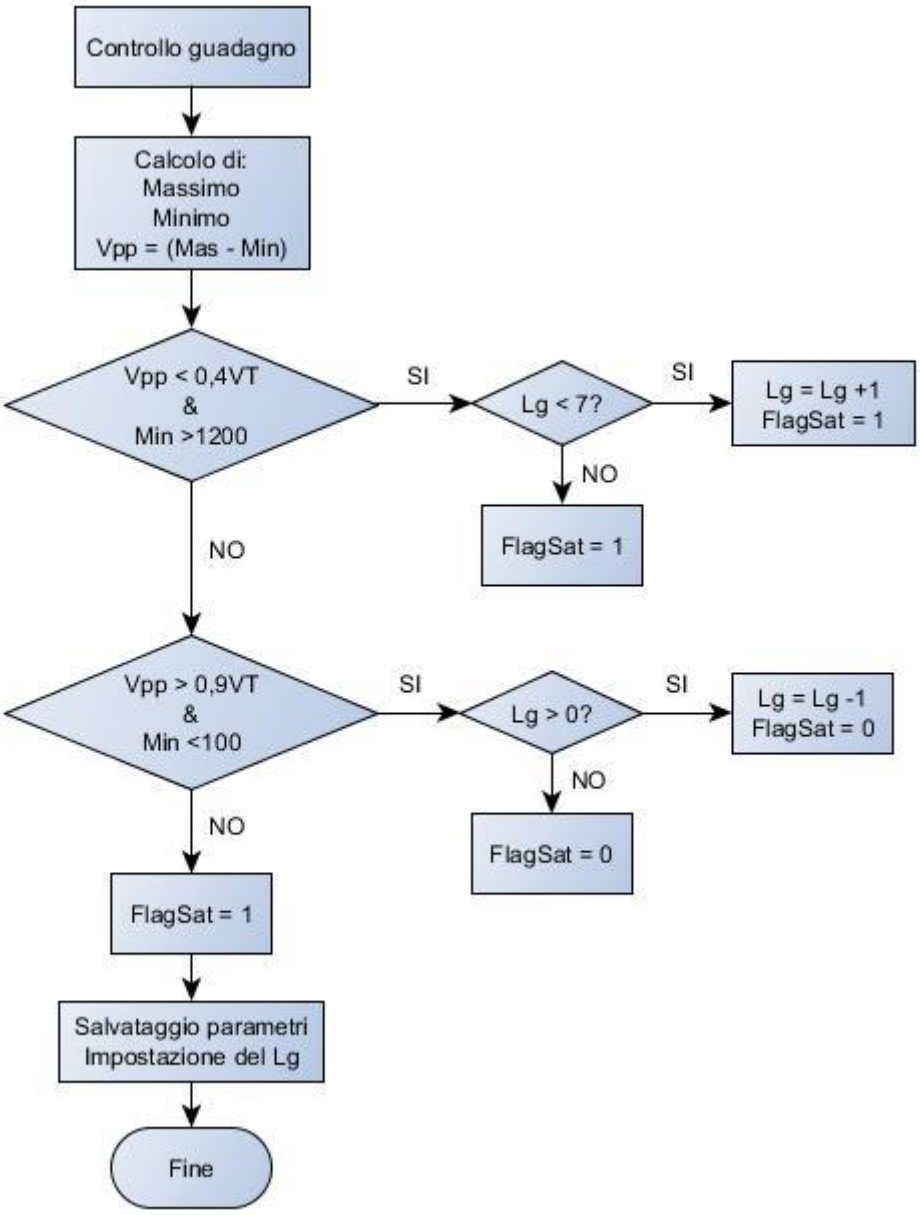


Figura 10: Controllo guadagno senza incmax

Este control se realiza por cada trama de muestras. Para el correcto análisis posterior o futuro de las muestras se debe saber la ganancia impuesta durante el muestreo, el canal y si hay que descartar la trama porque ha saturado la señal. Por esta razón, cada vez que se hace el control, hay que guardar estos tres datos de los cuales los dos últimos son los flags.

Cálculo de Vrms

Para obtener el valor de Vrms, se aplica la función:

$$V_{rms} = \lim_{T\to\infty} \sqrt{\frac{1}{T}\int_0^T [f(t)]^2 dt}$$

Como es una señal discreta en vez de continua, la formula usada es:

$$V_{rms} = \sqrt{\frac{\sum V^2 \cdot w^2}{300}}$$

donde V es el valor de la muestra y w el valore correspondiente de la ventana de peso. Como la entrada de Arduino es solo positiva, en realidad, la señal no está centrada en 0, tiene una componente de continua que debe ser eliminada. Además, el valor cuadrático medio obtenido de este modo se refiere al valor cuadrático medio de los valores de las muestras (entre 0 y 4096) , pero para tener los datos en unidades de tensión, es decir, en voltios, se deben transformar los datos .

Por último, el valor obtenido debe multiplicarse por un factor de 1,581 porque la ventana de pesaje atenúa la señal.

Al final, la fórmula usada es:

$$V_{rms} = 1{,}581 \cdot \frac{3{,}3}{4096} \sqrt{\frac{\sum (V - \bar{V})^2 \cdot w^2}{300}}$$

Entonces, el algoritmo es:

- Sumar todos los valores del vector, de la trama.
- Hacer la división de la suma entre el tamaño del vector para obtener el valor medio, es decir, aproximadamente el valor de la componente continua de la señal.
- Hacer el sumatorio: $\sum (V - \bar{V})^2 \cdot w^2$
- Hacer la raíz cuadrada del valor obtenido en el paso anterior.
- Finalmente se multiplica el último valor por el factor de corrección y por la fracción para tener el resultado en voltios.

Una vez obtenido el valor Vrms, se debe separar en parte real y la parte decimal para guardarlo en SD ya que la comunicación con la SD se debe realizar en Bytes.

Entonces, el diagrama de flujo del modo de calibración es el mostrado a continuación:

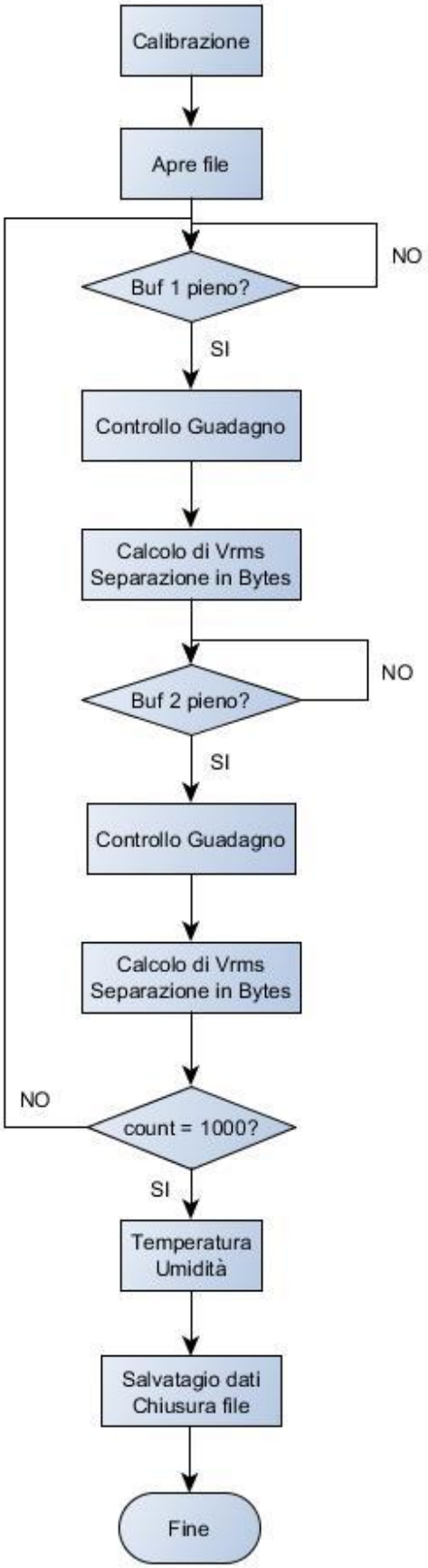


Figura 11: Calibrazione parametri

Cómo comprobación, al final de este proceso, se imprimen los datos almacenados en la pantalla. En la calibración no tiene que calcular la frecuencia fundamental, el propósito de hacerlo es encontrar la relación entre SPL y SAL, no caracterizar la señal o hacer cualquier otro análisis.